KATHOLIEKE
UNIVERSITEIT
LEUVEN

DEPARTEMENT TOEGEPASTE ECONOMISCHE WETENSCHAPPEN

RESEARCH REPORT 0255

INTERVAL SELECTION: APPLICATIONS, ALGORITHMS, AND LOWER BOUNDS

by
T. ERLEBACH
F. SPIEKSMA

D/2002/2376/55

Interval selection: Applications, algorithms, and lower bounds*

Thomas Erlebach
Computer Engineering and Networks Laboratory (TIK)
Eidgenössiche Technische Hochschule Zürich
erlebach@tik.ee.ethz.ch

Frits C.R. Spieksma
Faculty of Economic and Applied Economic Sciences
Katholieke Universiteit Leuven
frits.spieksma@econ.kuleuven.ac.be

November 13, 2002

Abstract

Given a set of jobs, each consisting of a number of weighted intervals on the real line, and a positive integer m, we study the problem of selecting a maximum weight subset of the intervals such that at most one interval is selected from each job and, for any point p on the real line, at most m intervals containing p are selected. This problem has applications in molecular biology, caching, PCB assembly, combinatorial auctions, and scheduling. It generalizes the problem of finding a (weighted) maximum independent set in an interval graph.

We give a parameterized algorithm GREEDY$_\alpha$ that belongs to the class of "myopic" algorithms, which are deterministic algorithms that process the given intervals in order of non-decreasing right endpoint and can either reject or select each interval (rejections are irrevocable). We show that there are values of the parameter α so that GREEDY$_\alpha$ produces a 2-approximation in the case of unit weights, an 8-approximation in the case of arbitrary weights, and a $(3+2\sqrt{2})$-approximation in the case where the weights of all intervals corresponding to the same job are equal. If all intervals have the same length, we prove for $m = 1, 2$ that GREEDY$_\alpha$ achieves ratio 6.638 in the case of arbitrary weights and 5 in the case of equal weights per job.

Concerning lower bounds, we show that for instances with intervals of arbitrary lengths, no deterministic myopic algorithm can achieve ratio better than 2 in the case of unit weights, better than ≈ 7.103 in the case of arbitrary weights, and better than $3+2\sqrt{2}$ in the case where the weights of all intervals corresponding to the same job are equal. If all intervals have the same length, we give lower bounds of $3+2\sqrt{2}$ for the case of arbitrary weights and 5 for the case of equal weights per job. Furthermore, we give a lower bound of $\frac{e}{e-1} \approx 1.582$ on the approximation ratio of randomized myopic algorithms in the case of unit weights.

*A preliminary version of some of the results in this paper has appeared in [10]. This research was partially supported by EU Thematic Network APPOL II, IST-2001-32007, with funding for the Swiss partners provided by the Swiss Federal Office for Education and Science (BBW).

1 Introduction

We study a *weighted job interval selection problem*, called WJISP. The input consists of jobs, each of which is given by a set of intervals on the real line, and a number m of available machines. (We use "WJISP$_m$" instead of "WJISP" if we want to make an explicit reference to the number m of machines.) Each interval i has a positive weight $w(i)$. A feasible solution is a subset of the given intervals such that (1) at most one interval is selected from each job, and (2) for any point p on the real line, at most m intervals overlapping p are selected. The goal is to find a feasible solution that maximizes the sum of the weights of the selected intervals. We let n denote the total number of intervals in the input. We assume that a sorted list of all interval endpoints is available; such a list can be constructed in time $O(n \log n)$.

Notice that the requirement that any point on the real line is overlapped by at most m selected intervals is equivalent to the requirement that the selected intervals can be partitioned into m subsets such that the intervals in each subset are pairwise disjoint. In some applications (see Section 2) this partition (in addition to specifying the selected intervals) is required as output. However, the subproblem of computing such a partition given the selected intervals can be solved efficiently by coloring the corresponding interval graph. In fact, n intervals can be colored in time $O(n)$ if the sorted list of interval endpoints is given. Therefore, we concentrate here on the problem of selecting the intervals and assume that, if required, an appropriate coloring procedure is employed to compute the partitioning.

WJISP$_1$ can be formulated in graph-theoretic terms. By constructing a graph such that there is a vertex for each interval, and two vertices are connected via an edge if the corresponding intervals overlap or if they belong to the same job, WJISP$_1$ can be viewed as a maximum-weight independent set problem in a graph. This graph is the edge union of an interval graph and a graph that consists of a disjoint union of cliques (cliques correspond to jobs).

There are several restricted versions of WJISP that are interesting (see the applications described in Section 2). We distinguish a number of variants of WJISP. Regarding the weights we consider:

- the *unweighted case* (called JISP), which refers to the case where each interval has the same weight w,
- *WJISP with equal weights per job*, which refers to instances of WJISP in which intervals that belong to the same job have the same weight, but intervals that belong to different jobs can have different weights, and finally
- *WJISP with arbitrary weights.*

Another interesting restriction pertains to the length of the intervals. One can distinguish in a similar fashion: *WJISP with equal lengths*, which refers to the case where each interval has the same length, *WJISP with equal lengths per job*, which refers to instances of WJISP in which intervals that belong to the same job have the same length, and finally *WJISP with arbitrary lengths.*

We consider a class of simple deterministic algorithms for WJISP and investigate so-called *worst-case ratios* (or approximation ratios) that can be obtained by algorithms within this class. Using standard terminology (see e.g. Hochbaum [17], Ausiello et al. [1]), we say that a deterministic algorithm for WJISP achieves (approximation) ratio ρ if it always outputs

a feasible solution whose weight is at least as large as $1/\rho$ times the weight of an optimal solution. A randomized algorithm achieves approximation ratio ρ if, on every instance of WJISP, the expected weight of the solution computed by the algorithm is at least $1/\rho$ times the weight of an optimal solution.

A natural class of algorithms to consider for WJISP instances is the class of *single-pass* algorithms. Very generally stated, single-pass algorithms are algorithms in which a feasible solution is obtained by iteratively making a decision concerning an item or an object. The first-fit decreasing algorithm for the bin packing problem and the nearest neighbor algorithm for the traveling salesman problem are prime examples of single-pass algorithms. This kind of algorithms can be useful since they need little computing time and/or little information (i.e., they can be applied in an on-line setting). In our context, we call an algorithm a single-pass algorithm when given some sequence of the intervals, each interval is (iteratively) either rejected or accepted (selected) without considering the intervals that will be processed later. Rejections are permanent, but an accepted interval can be rejected (preempted) at a later time. At any time, the set of currently selected intervals must be a feasible solution. After the last interval is presented, the set of currently selected intervals is taken as the solution computed by the algorithm.

When considering a specific single-pass algorithm for WJISP, it is crucial to specify the mechanism that determines the sequence in which the intervals will be processed. Different choices are possible, for instance processing the intervals in order of non-increasing weight, or processing the intervals in order of non-decreasing left endpoint or non-decreasing right endpoint. However, it is easy to see that single-pass algorithms that process the intervals in order of non-increasing weight or in order of non-decreasing left endpoint do not have a finite worst-case ratio (even in the case when each job consists of one interval only, see Woeginger [28], and even if randomization is allowed, see Canetti and Irani [6]). Therefore, we investigate in this paper the special class of single-pass algorithms, which we call *myopic algorithms*, that arise when the intervals are processed in order of non-decreasing right endpoint. Thus, myopic algorithms are deterministic single-pass algorithms that process the given intervals in order of non-decreasing right endpoint. These algorithms seem to be the simplest algorithms that achieve constant approximation ratios for WJISP. Let us emphasize here that we are primarily interested in the approximation ratios that can be achieved using single-pass algorithms, not in the (better) approximation ratios that can be achieved using arbitrary polynomial-time algorithms.

Analyzing myopic algorithms for WJISP can be seen as studying an on-line problem. We study the quality of the solutions that can be obtained by myopic algorithms. Using competitive analysis we show that for most settings that we investigate the algorithms proposed here are best possible, at least in the case $m = 1$.

In applications where the intervals correspond to time periods, an on-line scenario in which the algorithm receives the intervals in order of non-decreasing right endpoint may appear unnatural. Notice however that for instances where all intervals have the same length, the order of the left endpoints and the order of the right endpoints coincide. Therefore, the concept of myopic algorithms applies to the "real" on-line problem for such instances.

1.1 Known results

If each job consists of one interval only, WJISP_m reduces to finding a maximum weight m-colorable subgraph in an interval graph. This problem was shown to be solvable in polynomial

time by Yannakakis and Gavril [29]. If, in addition, $m = 1$ the problem reduces to finding a maximum weight independent set in an interval graph, for which a polynomial time algorithm was proposed by Frank [12].

The unweighted version of WJISP_1, called JISP_1, is studied by Spieksma [24]. It is shown that JISP_1 is MAX SNP-hard even if every job contains only two intervals and all intervals have the same length. Furthermore, it is shown that the value of the natural LP relaxation of JISP_1 is at most two times the value of an integral optimum, and a simple greedy algorithm that achieves approximation ratio 2 is presented.

A problem that is closely related to WJISP is the Time-Constrained Scheduling Problem (TCSP), see also Section 2. In that problem every job has a release time, a length, a deadline, and a weight. Viewed as a special case of WJISP with equal weights per job, every job in an instance of TCSP consists of all intervals of the required length between the release time and the deadline of the job. On the one hand the structure regarding the overlap of intervals of the same job is very restricted in TCSP, but on the other hand one must overcome the difficulty of dealing with an infinite number of intervals of a job (in particular, if release times, deadlines, and job lengths can be arbitrary real numbers). Bar-Noy et al. [3] give the following results for TCSP: for the unweighted case of TCSP_m, they give an iterative greedy algorithm (that does not belong to the class of single-pass algorithms) with approximation ratio $\rho(m) = \frac{(1+1/m)^m}{(1+1/m)^m-1}$. Note that $\rho(1) = 2$ and $\rho(m)$ tends to $e/(e-1) \approx 1.582$ as $m \to \infty$. For TCSP_m with equal weights per job a combinatorial algorithm called ADMISSION is described. ADMISSION consists of applying a greedy procedure m times. The time complexity of ADMISSION is $O(mn^2 \log n)$, where n is the number of jobs. An approximation ratio of $3 + 2\sqrt{2} \approx 5.828$ is proved for ADMISSION. Finally, for TCSP_m with equal weights per job, an LP-based algorithm is given that achieves a ratio of $\rho(m)$ (implying a 2-approximation algorithm for TCSP_1).

Some of the results described above can be generalized to WJISP_m with arbitrary weights. In particular, it is not difficult to verify that the LP-based algorithm and its analysis go through for our case, yielding a 2-approximation algorithm for WJISP_1 and a $\rho(m)$-approximation algorithm for WJISP_m. ADMISSION can be adapted to WJISP with equal weights per job as well, yielding a ratio of $3 + 2\sqrt{2} \approx 5.828$ for WJISP with equal weights per job. Notice that ADMISSION is not a myopic algorithm since it performs m passes over the given intervals. Improving the results of [3], Berman and DasGupta [4] and Bar-Noy et al. [2] proposed combinatorial two-phase algorithms that also achieve ratio 2 for WJISP_1 and, by repeated application, ratio $\rho(m)$ for WJISP_m. These algorithms do not belong to the class of single-pass algorithms. For the case of JISP, Chuzhoy et al. [8] improved the known ratios further and showed that for every $\varepsilon > 0$, there is a randomized approximation algorithm with ratio $e/(e-1) + \varepsilon$.

The on-line variant of TCSP is studied in Goldman et al. [15] and Goldwasser [16] in the single-machine case. The weight of a job is equal to its length and the algorithms receive the jobs in order of non-decreasing release times. Preemption is not allowed. In [15], a deterministic algorithm with ratio 2 if all jobs have the same length and a randomized algorithm with expected ratio $O(\log c)$ if the ratio of the longest to the shortest job length is c are presented. Note that "the special case of all jobs having the same length under the arbitrary delay model is of great interest" (quoted from [15]), e.g., for scheduling packets in an ATM switch (where all packets have the same length). In [16], better bounds are derived for the case that the slack of a job is at least proportional to its length.

Table 1: The results for WJISP with arbitrary lengths.

Variant of WJISP	Ratio of GREEDY$_\alpha$	Lower Bound
JISP	$2\ \forall m \geq 1$	$2\ \forall m \geq 1$
WJISP with equal weights per job	$3+2\sqrt{2} \approx 5.828\ \forall m \geq 1$	$3+2\sqrt{2} \approx 5.828 \quad m=1$
arbitrary weights	$8\ \forall m \geq 1$	$\approx 7.103 \quad m=1$

Table 2: The results for WJISP with equal lengths.

Variant of WJISP	Ratio of GREEDY$_\alpha$	Lower Bound
JISP	$2\ \forall m \geq 1$	$2\ \forall m \geq 1$
WJISP with equal weights per job	$5 \quad m=1,2$	$5 \quad m=1$
arbitrary weights	$\approx 6.638 \quad m=1,2$	$3+2\sqrt{2} \approx 5.828 \quad m=1$

1.2 Our results

Before describing our results let us first make the observation that WJISP$_m$ can be reduced to WJISP$_1$. This can be done by creating m disjoint copies of the original instance (every job of the new instance consists of all m copies of all its intervals in the original instance); this amounts to *projecting* the m machines onto disjoint parts of the real line. The implication of this observation is that WJISP$_m$ instances are special WJISP$_1$ instances or, in other words, any ρ-approximation algorithm for WJISP$_1$ provides a ρ-approximation algorithm for WJISP$_m$. This partly explains the phenomenon described in [3] that it seems that more machines allow better performance guarantees. Notice, however, that myopic algorithms cannot profit from this reduction: indeed a myopic algorithm processing the resulting instance of WJISP$_1$ would correspond to an algorithm that makes m passes over the intervals in the original instance.

The paper is organized as follows. In Section 2 we describe some applications of WJISP. In Section 3 we propose a myopic algorithm called GREEDY$_\alpha$ that can be implemented to run in $O(n^2)$ time (or in $O(n \log m)$ time if all intervals have the same length). Section 4 shows that with appropriate choices for the parameter α, GREEDY$_\alpha$ achieves the ratios described in Table 1 for WJISP with arbitrary lengths (the ratios for WJISP with equal lengths per job are the same) and in Table 2 for WJISP with equal lengths; each of these ratios is tight. Observe that GREEDY$_\alpha$ has the same ratio for WJISP with arbitrary lengths and equal weights per job as the (non-myopic) algorithm ADMISSION from [3], while having a lower time-complexity. Further, we prove in Section 5 that no myopic algorithm achieves better ratios than the ones described in Tables 1 and 2 under "Lower Bound". Our results show that GREEDY$_\alpha$ is optimal or close to optimal in the class of myopic algorithms. In Section 5.4 we prove that even a "randomized myopic" algorithm cannot achieve a ratio better than $\frac{e}{e-1} \approx 1.582$ for JISP. This shows that the use of randomization could at best improve the ratio for JISP from 2 to approximately 1.582. In Section 6 we state our conclusions. Appendix A gives the proof of Theorem 13.

2 Applications of WJISP

Interval scheduling problems have numerous applications and have been studied intensively (see for instance Fischetti et al. [11] and Kroon et al. [19]). In the following, we outline five concrete applications of WJISP.

PCB manufacturing (see Crama et al. [9]). In printed circuit board (PCB) manufacturing, preparing the production process requires placing so-called *feeders* on a feeder rack. The feeders deliver the components that are to be placed on prespecified locations on the board. Each feeder occupies a certain (small) number of consecutive slots in the rack. Every slot can hold at most one feeder. There are restrictions on the placement of feeders, i.e., each feeder can be placed only in a subset of all possible positions. Given a set of feeders, each with a list of admissible placements, it is desirable to place as many feeders as possible onto admissible positions on the feeder rack. If feeders differ in importance, it is also meaningful to assume that each feeder has a certain weight (priority) and to try to maximize the total weight of the feeders that are placed on the rack.

Viewed as an instance of WJISP_1, the feeders correspond to jobs and the admissible positions correspond to intervals. In this application, it is natural to assume that all intervals of a job have the same weight and the same length.

Molecular Biology (see Veeramachaneni et al. [27]). A fundamental problem in biology is to gain a better understanding of how functions are encoded in genes. An effective means of identifying functional regions in a genomic sequence is to compare it with the corresponding genomic region of another species. The Consensus Sequence Reconstruction (CSR) problem is encountered as a subproblem: given two sets $\mathcal{H}$ and $\mathcal{M}$ of DNA fragments (say, one set taken from human DNA and one set from mouse DNA), determine as much information about the order and orientation of the fragments as possible. Here, a fragment is just a sequence of symbols, each symbol corresponding to a conserved region. If $\mathcal{H}$ consists of a single (long) fragment H, the task is to align fragments in $\mathcal{M}$ with substrings of H such that different fragments are aligned with non-overlapping substrings. The goal is to maximize the sum of the alignment scores. Instances of this problem can be viewed as instances of WJISP_1 with arbitrary weights: each fragment in $\mathcal{M}$ corresponds to a job, and the substrings of H with which the fragment can be aligned are the intervals. The weight of an interval is the score of the corresponding alignment.

The same problem arises if $\mathcal{M}$ is not a set of fragments that can be aligned with substrings of H, but a set of hypotheses of the form "this region of H performs function x". For each hypothesis, there may be some substrings of H to which the hypothesis could apply. As a hypothesis may seem more plausible in one position than in another position, it is again meaningful to model this problem as WJISP_1 with arbitrary weights.

Combinatorial Auctions (see Rothkopf et al. [22]). Due to the ongoing sale of frequencies to providers of mobile telecommunications, and due to the ever-increasing popularity of e-commerce, the design of (combinatorial) auctions has become a popular research item. In a combinatorial auction different assets are for sale and bidders are allowed to bid for sets of assets. Given all bids from the bidders, a relevant problem is to decide what bids to accept in order to maximize total revenue (clearly, in general not all bids can be accepted since an asset can be sold at most once). In some cases the assets for sale posses a special structure, for instance when they can be linearly ordered. A popular example is the case where frequencies

are auctioned (indeed frequencies can be ordered by their magnitude), but other examples exist (see [22]). Suppose further that we allow only bids for sets of consecutive assets and allow at most one acceptance for each bidder (see [22, 18]). Then the problem of maximizing total revenue for this setting (the interval auction problem) becomes an instance of WJISP_1: a bidder is a job, their bids are the intervals and $m = 1$ since one can sell an asset at most once.

Let us now proceed to argue that the *design* of an interval auction can give rise to instances where the on-line interpretation of WJISP becomes relevant. Suppose, as described before, that the assets can be linearly ordered, say $1, 2, \ldots, T$. Moreover, the auction is designed in such a way that there are T rounds, and in each round t, asset t is added to the set of assets currently for sale (starting with the empty set). Again, as described before, bidders are only allowed to bid for sets of consecutive assets; moreover, in round t a bidder can only make a bid that includes asset t as the largest asset of the current bid. Of course, after each round, the bidders should receive information concerning what bids are currently active and what bids are currently rejected. Now, in principle it is possible to solve the resulting interval auction problem after round t to optimality. However, there are two arguments against such an approach. First, when playing many rounds, it may not be computationally feasible to compute the maximal revenue after each round t due to the intractability of the problem (see [24]). Second, a reasonable stipulation of such an auction would be that a bid that is rejected at some round cannot become "alive" again in later rounds (obviously this may happen when computing an optimal solution after each round). Thus, when designing an interval auction such that there is a round corresponding to each asset, we are faced with the on-line version of WJISP_1.

Caching (see Torng [26]). A cache is a small, fast memory that can temporarily store arbitrary memory items in order to allow the CPU to access them faster than in main memory. For the purpose of analyzing cache performance, we view the execution of a program as a sequence of accesses to memory items. When a memory item x is accessed and does not yet reside in the cache, it can be (but doesn't have to be; we allow cache bypassing) brought into the cache. If x is still in the cache when it is accessed a second time later on, this is called a *cache hit*. With every cache hit the cost for an expensive access to main memory is avoided. We view the period between two accesses to the same item x as an interval on the real line. At the time of an access, at most one interval ends and at most one new interval begins. Selecting an interval i means that x is brought into (or remains in) the cache at the access to x at the beginning of i and stays in the cache until the access to x at the end of i, thus yielding a cache hit. If every memory item can go into an arbitrary cache location (i.e., if we have a fully associative cache) and if the cache has m locations, the problem of maximizing the cache hits can be viewed as an instance of WJISP_m where each job consists of a single interval. In this application, it seems natural to assume that the intervals are unweighted, but there may be other factors that make it more desirable to achieve cache hits for certain accesses, thus giving instances of WJISP_m with equal weights per job. If two consecutive intervals between accesses to x are selected, an additional constraint is that x must reside in the same cache location during both intervals; otherwise, we would have to assume that x can move from one cache location to another at no cost. However, this additional constraint can easily be satisfied in the procedure that colors the interval graph corresponding to the selected intervals.

Due to the high hardware cost for fully associative caches, t-way set associative caches are often used instead in practice. Here, a cache with k locations is partitioned into t direct

mapped caches, each of size k/t. A memory item x is mapped to a position $p(x)$, $1 \leq p(x) \leq k/t$, and can be stored only in location $p(x)$ in each of the t direct mapped caches. Conceptually, this can be viewed as partitioning the cache into k/t sub-caches, each of size t, such that each sub-cache is fully associative and such that every memory item can go in only one of the k/t sub-caches. The problem of maximizing cache hits in a t-way set associative cache with k locations can thus be solved by solving the subproblems for each of the k/t sub-caches independently. In terms of WJISP, this amounts to k/t disjoint instances of WJISP_t that can be combined into a single instance by projecting them onto disjoint parts of the real line.

Finally, consider the case that there can be more general restrictions on the cache locations available to a memory item (e.g., as in t-way skewed associative caches [23]). For every memory item x, there is a number of admissible locations in the cache where the item can be stored. The problem of maximizing cache hits can then be modeled as an instance of WJISP_1 as follows: project the timelines of all cache locations onto disjoint parts of the real line and add, for every period between consecutive accesses to x, an interval in those parts of the real line that correspond to cache locations that are admissible for x (all intervals for this period belong to one job). However, it should be noted that the constraint that items cannot move within the cache is ignored by this approach.

Time-Constrained Scheduling (see Bar-Noy et al. [3]). Consider the following scheduling problem: we are given m machines (identical or unrelated) and n tasks, and each task has a release time, a deadline, a processing time (that can depend, in the case of unrelated machines, on the machine on which the task is executed), and a weight. We want to select a subset of the given tasks and schedule them on the machines non-preemptively such that every selected task is scheduled no earlier than its release time and finishes no later than its deadline. The goal is to maximize the sum of the weights of the scheduled tasks.

We can view this scheduling problem for m identical machines as an instance of WJISP_m with equal weights per job (tasks correspond to jobs, and every possible execution of a task corresponds to an interval) and the problem for m unrelated machines as an instance of WJISP_1 (by projecting the timelines of all m machines onto different parts of the real line). Notice that the intervals in instances of WJISP arising from this application display a special structure. If all release times, deadlines, and processing times are integers that are bounded by a polynomial in the size of the input (i.e., if we have *polynomially bounded integral input*), the resulting instances of WJISP_m and WJISP_1 have size polynomial in the original instance (only intervals with integral starting times must be considered).

3 Algorithm GREEDY_α

We propose a myopic algorithm called GREEDY_α, shown in Figure 1, as an approximation algorithm for WJISP. It has a parameter α that can take (meaningful) values in the range $[0, 1]$. GREEDY_α considers the intervals in order of non-decreasing right endpoint. It maintains a set S of currently selected intervals. When it processes an interval i, it computes a set $C_i \subseteq S$ such that i could be selected after preempting the intervals in C_i and such that C_i has minimum weight among all such sets. C_i is called the *cheapest conflict set* for i. The algorithm selects i only if $w(C_i) \leq \alpha w(i)$, i.e., if the total weight of selected intervals increases by at least $(1 - \alpha)w(i)$ if i is selected and the intervals from C_i are preempted.

Let us briefly compare GREEDY_α to the algorithm ADMISSION of [3], which was found

Algorithm GREEDY$_\alpha$

$S = \emptyset$; { set of currently accepted intervals }
for all intervals, in order of non-decreasing right endpoint **do**
    i = current interval;
    C_i = minimum-weight subset of S such that $(S \setminus C_i) \cup \{i\}$ is feasible;
    if $w(C_i) \leq \alpha w(i)$ **then**
        $S = (S \setminus C_i) \cup \{i\}$;
    fi;
od;
return S;

Figure 1: Algorithm GREEDY$_\alpha$.

independently of our work. The basic spirit of the algorithms is similar: For the case of WJISP$_1$ with equal weights per job, the two algorithms are essentially identical, and the parameters α of GREEDY$_\alpha$ and β of ADMISSION are related by $\alpha = 1/\beta$. For the case $m > 1$, however, ADMISSION passes through the intervals m times (once for each machine) [3], while our algorithm GREEDY$_\alpha$ is a single-pass algorithm for any value of m. Furthermore, GREEDY$_\alpha$ can deal with the case of arbitrary weights, while ADMISSION is specified only for the case of equal weights per job.

We are interested in an efficient implementation of GREEDY$_\alpha$. In Section 3.1, we show how the cheapest conflict set C_i can be determined efficiently, which gives rise to a total running time of $O(n^2)$ of GREEDY$_\alpha$. As a byproduct of this subsection we show how knowing that an interval graph is m-colorable gives you an $O(n)$ algorithm for obtaining a maximum-weight $(m-1)$-colorable subgraph as compared to $O(mS(n))$ in the general case [7], where $S(n)$ denotes the running time for any algorithm for finding a shortest path in a directed graph with $O(n)$ arcs and positive arc weights. (With an efficient implementation of Dijkstra's algorithm, for example, $S(n)$ can be taken as $O(n \log n)$. There are algorithms that are asymptotically faster (see, e.g., [13, 21] and further references given in [25]); however, these algorithms seem of theoretical interest only and do not achieve a linear running-time. The only linear-time shortest paths algorithm known so far is due to Thorup [25]; it works for undirected graphs.) In Section 4, we analyze the approximation ratio achieved by GREEDY$_\alpha$.

3.1 Determining the cheapest conflict set

Let i be the interval that is currently processed by GREEDY$_\alpha$. In the special case of $m = 1$, the set C_i is simply the set of all intervals in S that intersect i and, possibly, an interval $i' \in S$ that belongs to the same job as i. Therefore, C_i can be determined easily in this case. In the following, we show how to deal with the more complicated case of arbitrary m.

If S contains an interval i' that belongs to the same job as i, it is clear that C_i must contain i'. In that case, let $C_i' = C_i \setminus \{i'\}$, otherwise let $C_i' = C_i$. Let $Q_i \subseteq S$ be the subset of currently selected intervals that intersect the interval i and that do not belong to the same job as i. See Figure 2. Obviously, C_i' is a minimum-weight subset of Q_i such that $Q_i \setminus C_i'$ is $(m-1)$-colorable. Hence, the set $Q_i \setminus C_i'$ is a maximum-weight $(m-1)$-colorable subset of Q_i. Thus, the problem of determining the cheapest conflict set is equivalent to the problem of finding a maximum weight $(m-1)$-colorable subgraph in an m-colorable interval graph.

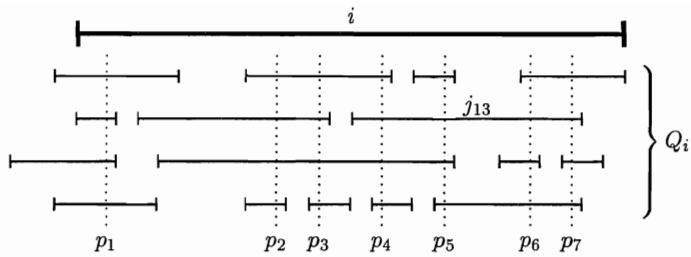


Figure 2: Current interval i and intersecting intervals in S.

```
C(0) = 0;
for h = 1 to r do C(h) = ∞; od;
for ℓ = 1 to s do
    if first(j_ℓ) is not undefined then
        for h = first(j_ℓ) to last(j_ℓ) do
            C(h) = min{C(h), C(first(j_ℓ) − 1) + w(j_ℓ)};
        od;
    fi;
od;
```

Figure 3: Dynamic programming algorithm.

Therefore, $Q_i \setminus C_i'$ and C_i' could be determined in polynomial time using an algorithm that solves the maximum-weight k-colorable subgraph problem in interval graphs for arbitrary values of k. However, since our problem is more specific (we know that Q_i is m-colorable), we can compute the set C_i' in time $O(n)$ using a dynamic programming approach, as witnessed by the following theorem.

Theorem 1 *The cheapest conflict set can be computed in $O(n)$ time.*

Proof: For every interval $j \in Q_i$, consider a point p just before the right endpoint of j. If p is contained in m intervals of Q_i, at least one of these intervals must be in C_i'. Let $p_1, \ldots, p_r$ be all such points (i.e., points just before the right endpoint of an interval in Q_i) that are contained in m intervals of Q_i. It is clear that C_i' is a minimum-weight subset of Q_i such that every point p_h, $1 \leq h \leq r$, is covered by that subset. (We say that a set of intervals *covers* a point if the point is contained in at least one interval of that set.)

Let $j_1, j_2, \ldots, j_s$ denote the intervals in Q_i in order of non-decreasing right endpoint. For every interval $j_\ell \in Q_i$, let $first(j_\ell)$ and $last(j_\ell)$ be the smallest resp. largest index of a point p_h that is contained in j_ℓ. (If j_ℓ does not contain any point p_h, $first(j_\ell)$ and $last(j_\ell)$ are undefined.) In Figure 2, we have $first(j_{13}) = 4$ and $last(j_{13}) = 7$, for example. The points $p_1, \ldots, p_r$, the intervals $j_1, \ldots, j_s$, and the tables *first* and *last* can all be constructed in time $O(n)$. We use the dynamic programming procedure shown in Figure 3 to compute values $C(h)$, $1 \leq h \leq r$.

Claim 2 *After ℓ iterations of the outer for-loop, it holds for $1 \leq h \leq r$ that $C(h)$ is the weight of a minimum-weight subset of $\{j_1, \ldots, j_\ell\}$ that covers $p_1, \ldots, p_h$ (or ∞ if no such subset exists).*

Obviously, Claim 2 implies that $C(r) = w(C'_i)$ at the end of the execution of the procedure, and additional bookkeeping will allow to construct C'_i at no extra cost. Thus, the correctness of the algorithm follows from Claim 2, which can be proved easily by induction on ℓ. The running-time of the procedure sketched in Figure 3 is $O(s + rm) = O(nm)$, because the body of the inner for-loop is executed exactly rm times. (For each of the r points p_h, the body of the inner for-loop is executed only for the m intervals containing p_h.)

Let us now describe an implementation of the algorithm in Figure 3 that runs in $O(n)$ time. To begin with, notice that the costs $C(h)$, $0 \leq h \leq r$, are non-decreasing with h at any stage of the algorithm. We will maintain disjoint consecutive subsets of $\{0, 1, \ldots, r\}$ with the property that for each subset X, the value $C(h)$ is the same for all $h \in X$. Elements with $C(h) = \infty$ are kept in singleton subsets. For maintaining the subsets, we use the data structure due to Gabow and Tarjan [14]. We may assume that this data structure provides the following operations:

- INITIALIZE(r): initialize the data structure with $r + 1$ singleton sets $\{0\}$, $\{1\}, \ldots, \{r\}$.
- FIND(x): return the first and last element of the subset that currently contains x, i.e., if x is currently in the set $\{a, a + 1, \ldots, b\}$, then return the pair (a, b).
- UNION(x, y): merge the sets containing x and y. Precondition: x and y are in different sets, x is the largest element in the set containing x, and y is the smallest element in the set containing y.

The precondition of the UNION operation ensures that the structure of all potential UNION operations is a chain and thus the data structure of [14] is applicable (that data structure requires that the potential UNION operations form a tree). The total running-time for the initialization of the data structure and a sequence consisting of $O(s)$ operations FIND(x) and up to r operations UNION(x, y) is then $O(r + s)$.

For a subset X maintained by the data structure, we store the common value $C(h)$ for all $h \in X$ with the largest element in X (the second component of the pair returned by FIND(h) for any $h \in X$). For an element h that is not the largest element of its subset, the value of the variable $C(h)$ can be arbitrary.

The pseudo-code of the resulting implementation is shown in Figure 4. For each interval j_ℓ, the algorithm first determines the value $C(\mathit{first}(j_\ell)-1)$ by executing $(a, b) =$ FIND($\mathit{first}(j_\ell)-1$) and then accessing $C(b)$. This allows to compute the value $w = C(b) + w(j_\ell)$, which is the cost of the new candidate set that covers all p_i with $i \leq \mathit{last}(j_\ell)$ and has j_ℓ as its rightmost interval. If the elements in the set containing $\mathit{last}(j_\ell)$ have a cost larger than w, their cost is updated to w by setting $C(d) = w$. (Note that the set containing $\mathit{last}(j_\ell)$ must have $\mathit{last}(j_\ell)$ as its largest element, because all elements $h > \mathit{last}(j_\ell)$ still have cost $C(h) = \infty$.) In this case, the algorithm then checks repeatedly whether the elements in the set that is just before the set containing $\mathit{last}(j_\ell)$ have a cost larger than w and, if so, merges that set with the set containing $\mathit{last}(j_\ell)$. The effect of each such UNION operation is that the elements h in the set with smaller elements implicitly receive the same value $C(h) = w$ as the elements in the set containing $\mathit{last}(j_\ell)$.

```
INITIALIZE(r);
C(0) = 0;
for h = 1 to r do C(h) = ∞; od;
for ℓ = 1 to s do
    if first(j_ℓ) is not undefined then
        (a, b) = FIND(first(j_ℓ) − 1);
        w = C(b) + w(j_ℓ);
        (c, d) = FIND(last(j_ℓ)); /* will give d = last(j_ℓ) */
        if C(d) > w then
            C(d) = w;
            (e, f) = FIND(c − 1); /* will give f = c − 1 */
            while C(f) > w do
                UNION(f, c);
                c = e;
                (e, f) = FIND(c − 1); /* will give f = c − 1 */
            od;
        fi;
    fi;
od;
```

Figure 4: Linear-time implementation of the dynamic programming algorithm.

It is not difficult to see that this is a correct implementation of the dynamic programming algorithm of Figure 3. It remains to analyze the running-time. For each interval j_ℓ, the number of FIND operations executed in the body of the loop is $2+k$, where k is the number of iterations of the inner while loop. Since each iteration of the inner while loop executes a UNION operation and there can be at most r such operations, the total number of FIND operations can be bounded by $2s + r$. Hence, the total running-time of the algorithm is $O(s + r) = O(n)$. Moreover, the cheapest conflict set itself can again be computed easily in the same running-time by storing with each value $C(h)$ also the index of the rightmost interval of the solution covering $p_1, \ldots, p_h$ that has cost $C(h)$. □

Theorem 1 leads to the following corollaries:

Corollary 3 *A maximum weight $(m-1)$-colorable subgraph in an m-colorable interval graph, given by the sorted list of interval endpoints, can be obtained in $O(n)$ time.*

Corollary 4 *GREEDY$_\alpha$ runs in $O(n^2)$ time.*

For WJISP with equal lengths, computing the cheapest conflict set is much easier, since at most m intervals in S can overlap the current interval i. The cheapest conflict set C_i contains the interval in S that belongs to the same job as i (if such an interval exists) as well as the cheapest interval overlapping i (if there are m intervals in S that overlap i and belong to a different job). We maintain a balanced search tree T such that, if the current interval has left endpoint p, T stores all intervals in S overlapping p, sorted by their weights. Determining the cheapest conflict set and updating T takes time $O(\log m)$, giving a total running time of $O(n \log m)$ for GREEDY$_\alpha$ in the case of WJISP with equal lengths.

4 Analysis of approximation ratio

In this section, we give tight bounds on the approximation ratio of GREEDY$_\alpha$ for the different variants of WJISP. The section is divided into two parts: in Section 4.1 we investigate the case where intervals have arbitrary lengths, and in Section 4.2 we deal with the case where all intervals have the same length. Each subsection deals with the three possibilities concerning the weights of the intervals: equal weights, equal weights per job, and arbitrary weights.

Let us now introduce some notation. We use A to denote the set of intervals that is returned by GREEDY$_\alpha$, we use T for the set of intervals that were selected at least at some time by GREEDY$_\alpha$, and we use OPT for some set of intervals that constitutes an optimal solution. Their values are referred to as $w(A), w(T)$ and $w(OPT)$, respectively. Further, as mentioned before, the set S is the set of selected intervals at some point during the execution of the algorithm. The basic idea of the analysis is to charge the weight of the intervals in an optimal solution to the intervals selected by GREEDY$_\alpha$ (the set T) and next to derive bounds on the amount of charge received by intervals in A. An inequality that is fundamental in the analysis is:

$$w(A) \geq (1-\alpha)w(T) \tag{1}$$

This inequality holds because GREEDY$_\alpha$ selects a new interval i only if the total weight of currently selected intervals increases by at least $(1-\alpha)w(i)$. Thus, first we bound $w(OPT)$ in terms of $w(T)$ and, using (1), in terms of $w(A)$.

4.1 WJISP with arbitrary lengths

Here, we analyze the approximation ratio of GREEDY$_\alpha$ in the case that the given intervals have arbitrary lengths. For the case of arbitrary weights, we have the following theorem.

Theorem 5 *[arbitrary lengths] For WJISP$_m$ with arbitrary weights, GREEDY$_\alpha$ achieves approximation ratio $\frac{2}{\alpha(1-\alpha)}$. This result is tight even in the case of equal lengths per job.*

Proof: Consider an interval $i \in OPT$. If $i \in T$, we charge $w(i)$ to i. If $i \in OPT \setminus T$, consider the instant when GREEDY$_\alpha$ processed interval i. Let C_i denote the minimum-weight set of intervals whose removal from S would have allowed to accept i. If S contains an interval from the same job as i, denote that interval by i'; otherwise, let i' be an imaginary interval with zero weight (just to simplify the formulas).

Let Q_i denote the set of all intervals in S that intersect i and that do not belong to the same job as i. As S is feasible, Q_i can be partitioned into m sets $Q_{i1}, \ldots, Q_{im}$ of intervals such that the intervals in each set $Q_{i\ell}$ are pairwise disjoint. Note that

$$w(i') + w(Q_{i\ell}) > \alpha w(i) \text{ for } 1 \leq \ell \leq m, \tag{2}$$

because of the definition of C_i and because GREEDY$_\alpha$ did not accept i.

We charge $\min\{w(i), \frac{1}{\alpha}w(i')\}$ to i'. If the remaining weight $w(i) - \min\{w(i), \frac{1}{\alpha}w(i')\}$ is positive, we divide it into m equal parts and distribute each part among the intervals in one set $Q_{i\ell}$ such that an interval $j \in Q_{i\ell}$ is charged

$$\frac{w(i) - \min\{w(i), \frac{1}{\alpha}w(i')\}}{m} \cdot \frac{w(j)}{w(Q_{i\ell})}.$$

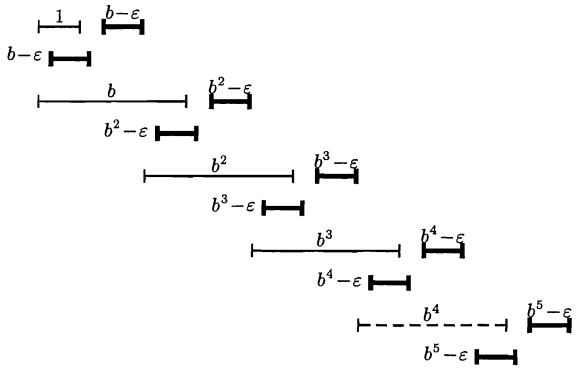


Figure 5: Worst-case example with arbitrary weights.

(2) implies that every interval $j \in Q_i$ is charged by i for at most $\frac{1}{\alpha m} w(j)$ in this way. Altogether, every interval $j \in T$ can receive charge at most $\frac{2}{\alpha} w(j)$ (charge at most $\frac{1}{\alpha} w(j)$ from an interval belonging to the same job and charge at most $\frac{1}{\alpha} w(j)$ from intervals in OPT that overlap the right endpoint of j), implying that $w(OPT) \leq \frac{2}{\alpha} w(T) \leq \frac{2}{\alpha(1-\alpha)} w(A)$.

To see that our analysis is tight, consider the instance of WJISP$_1$ shown in Figure 5. Intervals in the same row belong to the same job, and the labels correspond to the weights of the intervals, where $b = \frac{1}{\alpha}$ and ε is a very small positive value. Every group of three intervals is called a *phase*; the shown example consists of 5 phases. In an example with k phases, GREEDY$_\alpha$ will output a single interval of weight b^{k-1} (shown dashed in Figure 5), while the optimal solution contains $2k$ intervals (shown bold in Figure 5) of total weight $\sum_{i=0}^{k-1} 2(b^{i+1} - \varepsilon) = 2b\frac{b^k-1}{b-1} - 2k\varepsilon$. Simple calculations show that the approximation ratio of GREEDY$_\alpha$ tends to $\frac{2}{\alpha(1-\alpha)}$ as ε goes to zero and k goes to infinity.

This construction can be adapted to WJISP$_m$ for $m > 1$ by including m copies of every interval (putting each copy in a different job). Furthermore, the intervals can be stretched such that all intervals belonging to the same job have the same length, without changing the optimal solution or the solution produced by GREEDY$_\alpha$. □

Corollary 6 *[arbitrary lengths] For WJISP$_m$ with arbitrary weights, GREEDY$_\alpha$ performs best for $\alpha = \frac{1}{2}$, in which case it achieves approximation ratio* 8.

In the cases of unit weights and of equal weights per job, the proof technique of Theorem 5 can be adapted to obtain the following results.

Theorem 7 *[arbitrary lengths] For JISP$_m$, GREEDY$_\alpha$ achieves approximation ratio* 2 *for any α in the range $[0, 1)$. This result is tight even in the case where all intervals have the same length.*

Proof: Since all intervals have the same weight, GREEDY$_\alpha$ never preempts a selected interval and we have that $T = A$ (provided that $0 \leq \alpha < 1$).

Consider an interval $i \in OPT$. If $i \in A$, we charge $w(i)$ to i. If $i \in OPT \setminus A$, consider the time when GREEDY$_\alpha$ processed interval i. If S contained an interval j belonging to the

same job as i at that time, charge $w(i)$ to j. If S did not contain an interval belonging to the same job as i, let Q_i be the set of intervals that were contained in S and that intersected i. As S is feasible, Q_i can be partitioned into m sets $Q_{i1}, \ldots, Q_{im}$ of intervals such that the intervals in each set $Q_{i\ell}$ are pairwise disjoint. As GREEDY$_\alpha$ did not accept i, each set $Q_{i\ell}$ must be non-empty. We charge $w(i)/m$ to an arbitrary interval from each set $Q_{i\ell}$.

As every interval $j \in A$ receives charge at most $w(j)$ from intervals of the same job and charge at most $m \cdot w(j)/m$ from intervals overlapping the right endpoint of j, we have that $w(OPT) \leq 2w(A)$.

The tightness of ratio 2 follows from the lower bound of 2 on the ratio of any myopic algorithm for JISP$_m$, which we will give in Theorem 22. □

Theorem 8 *[arbitrary lengths] For WJISP$_m$ with equal weights per job, GREEDY$_\alpha$ achieves approximation ratio $\frac{1+\alpha}{\alpha(1-\alpha)}$. This result is tight even in the case of equal lengths per job.*

Proof: Consider an interval $i \in OPT$. If $i \in T$, we charge $w(i)$ to i. If $i \in OPT \setminus T$, consider the time when GREEDY$_\alpha$ processed interval i. If S contained an interval j belonging to the same job as i at that time, charge $w(i)$ to j. Now assume that S did not contain such an interval j. Let C_i denote the minimum-weight set of intervals whose removal from S would have allowed to accept i.

Let Q_i denote the set of all intervals in S that intersect i. Note that no interval in Q_i belongs to the same job as i. As S is feasible, Q_i can be partitioned into m sets $Q_{i1}, \ldots, Q_{im}$ of intervals such that the intervals in each set $Q_{i\ell}$ are pairwise disjoint. Note that

$$w(Q_{i\ell}) > \alpha w(i) \text{ for } 1 \leq \ell \leq m, \tag{3}$$

because of the definition of C_i and because GREEDY$_\alpha$ did not accept i. We divide $w(i)$ into m equal parts and distribute each part among the intervals in one set $Q_{i\ell}$, where the charge one interval in $Q_{i\ell}$ receives is proportional to the weight of that interval. More precisely, interval $j \in Q_{i\ell}$ is charged

$$\frac{w(i)}{m} \cdot \frac{w(j)}{w(Q_{i\ell})}.$$

We can deduce from (3) that every interval $j \in Q_i$ is charged by i for at most $\frac{1}{\alpha m} w(j)$ in this way. It follows that every interval $j \in T$ is charged at most $w(j)$ by intervals from the same job and at most $\frac{1}{\alpha} w(j)$ by other intervals (which must overlap the right endpoint of j). Therefore, we get $w(OPT) \leq (1 + \frac{1}{\alpha}) w(T) \leq \frac{1+\alpha}{\alpha(1-\alpha)} w(A)$.

To see that our analysis is tight, consider the instance of WJISP$_1$ with equal weights per job shown in Figure 6. Intervals in the same row belong to the same job, and the labels correspond to the weights of the intervals, where $b = \frac{1}{\alpha}$ and ε is a very small positive value. Every group of three intervals is called a *phase*; the shown example consists of 5 phases. In an example with k phases, GREEDY$_\alpha$ will output a single interval of weight b^{k-1} (shown dashed in Figure 6), while the optimal solution contains $2k$ intervals (shown bold in Figure 6) of total weight $\sum_{i=0}^{k-1} (b^i + b^{i+1} - \varepsilon) = (1+b)\frac{b^k-1}{b-1} - k\varepsilon$. Simple calculations show that the approximation ratio of GREEDY$_\alpha$ tends to $\frac{1+\alpha}{\alpha(1-\alpha)}$ as ε goes to zero and k goes to infinity.

This construction can be adapted to WJISP$_m$ for $m > 1$ by including m copies of every interval (putting each copy in a different job). Furthermore, the intervals can be stretched such that all intervals belonging to the same job have the same length, without changing the optimal solution or the solution produced by GREEDY$_\alpha$. □

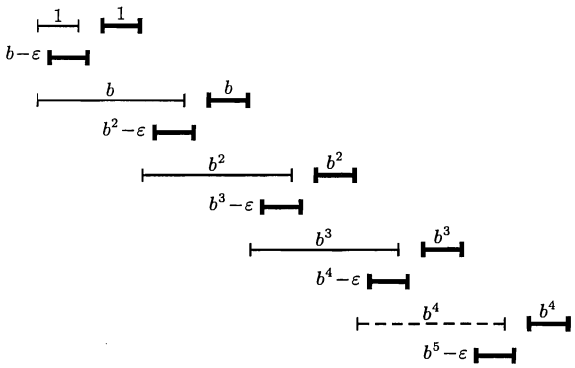


Figure 6: Worst-case example with equal weights per job.

Corollary 9 *[arbitrary lengths] For $WJISP_m$ with equal weights per job, $GREEDY_\alpha$ performs best for $\alpha = \sqrt{2} - 1 \approx 0.414$, in which case it achieves approximation ratio $3 + 2\sqrt{2} \approx 5.828$.*

Note that this result is consistent with the result of Bar-Noy et al. [3], who proved independently that ADMISSION performs best for TCSP with parameter $\beta = 1 + \sqrt{2}$ and achieves ratio $3 + 2\sqrt{2}$ in this case (their parameter β corresponds to $1/\alpha$ in our setting).

4.2 WJISP with equal lengths

Now we consider WJISP in the case where all intervals of all jobs have the same length. In applications where the intervals correspond to time intervals, myopic algorithms are "real" on-line algorithms in this case, as the order of left endpoints and the order of right endpoints coincide.

For JISP with equal lengths, the approximation ratio of GREEDY_α is 2, the same as in the case of arbitrary lengths (Theorem 7), and this is again tight. Therefore, we need to consider here only WJISP with arbitrary weights (Section 4.2.1) and WJISP with equal weights per job (Section 4.2.2). It turns out that for these cases GREEDY_α achieves better ratios for equal lengths than for arbitrary lengths. The analysis, however, gets a bit more complicated, and we do not have tight results for all values of m.

4.2.1 Arbitrary weights

We consider WJISP with equal lengths and arbitrary weights. First, we consider the case $m = 1$.

Theorem 10 *[equal lengths] In the case of $WJISP_1$ with arbitrary weights, $GREEDY_\alpha$ achieves approximation ratio $\frac{2+\alpha}{\alpha(1-\alpha^2)}$.*

Proof: We distinguish *job charge* and *overlap charge*. Consider an interval $i \in OPT$ and the instant when GREEDY_α processed that interval. If $i \in T$, charge $w(i)$ to i and call this charge *overlap charge*. Now assume that $i \notin T$. Let Q_i be the set of intervals in S that are in conflict with i when GREEDY_α processes i. Note that Q_i contains at most one interval

intersecting i (because all intervals have the same length) and at most one additional interval belonging to the same job as i. Therefore, we have $|Q_i| \leq 2$. Each interval $j \in Q_i$ is charged

$$\frac{w(j)}{w(Q_i)} \cdot w(i) \ .$$

As GREEDY$_\alpha$ did not select i, we have $w(Q_i) > \alpha w(i)$, so each interval $j \in Q_i$ receives charge at most $w(j)/\alpha$ from i. If j intersects i, we call the charge that j receives from i *overlap charge*, otherwise *job charge*.

In order to facilitate a tight analysis, we redistribute some of the job charge. Consider an interval $j \in T$ that receives job charge from an interval $i \in OPT$ belonging to the same job. If $j \in A$, we do nothing. If $j \in T \setminus A$, this means that j was preempted by GREEDY$_\alpha$ at some later time in favor of an interval k belonging to the same job as j. Observe that k cannot have received any job charge, as the interval $i \in OPT$ that belongs to the same job as j and k was processed by GREEDY$_\alpha$ before k. Now we redistribute the total charge that j and k have received, which is bounded from above by $\frac{2}{\alpha}w(j) + \frac{1}{\alpha}w(k)$, proportionally among these two intervals. Since $w(k) \geq w(j)/\alpha$, we can bound the resulting charge c_j for interval j as follows:

$$\begin{aligned} c_j &\leq \frac{w(j)}{w(j)+w(k)} \left(\frac{2}{\alpha}w(j) + \frac{1}{\alpha}w(k)\right) = w(j)\left(\frac{1}{\alpha} + \frac{\frac{1}{\alpha}w(j)}{w(j)+w(k)}\right) \\ &\leq w(j)\left(\frac{1}{\alpha} + \frac{1}{\alpha(1+\frac{1}{\alpha})}\right) = w(j)\left(\frac{1}{\alpha} + \frac{1}{\alpha+1}\right) \end{aligned}$$

In the same way, we obtain that the resulting charge c_k for interval k is bounded by

$$c_k \leq w(k)\left(\frac{1}{\alpha} + \frac{1}{\alpha+1}\right) \ .$$

Putting everything together, we get that every interval $j \in A$ receives charge at most $\frac{2}{\alpha}w(j)$, while every interval $k \in T \setminus A$ receives charge at most $(\frac{1}{\alpha} + \frac{1}{\alpha+1})w(k)$. We obtain:

$$\begin{aligned} w(OPT) &\leq w(A) \cdot \frac{2}{\alpha} + w(T \setminus A)\left(\frac{1}{\alpha} + \frac{1}{\alpha+1}\right) \\ &= w(T)\left(\frac{1}{\alpha} + \frac{1}{\alpha+1}\right) + w(A)\left(\frac{1}{\alpha} - \frac{1}{\alpha+1}\right) \\ &\leq w(A)\frac{1}{1-\alpha}\left(\frac{1}{\alpha} + \frac{1}{\alpha+1}\right) + w(A)\left(\frac{1}{\alpha} - \frac{1}{\alpha+1}\right) \\ &= w(A)\frac{2+\alpha}{\alpha(1-\alpha^2)} \end{aligned}$$

□

This result is tight as witnessed by the following theorem:

Theorem 11 *[equal lengths] For every $m \geq 1$, the approximation ratio of GREEDY$_\alpha$ for WJISP$_m$ with arbitrary weights is not better than $\frac{2+\alpha}{\alpha(1-\alpha^2)}$.*

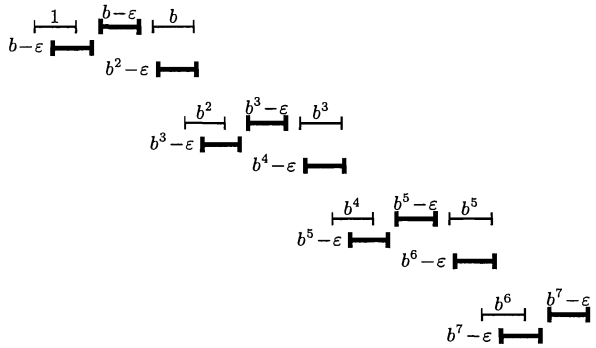


Figure 7: Worst-case example for equal lengths and arbitrary weights

Proof: The construction of worst-case examples for WJISP_1 is illustrated in Figure 7, where again $b = \frac{1}{\alpha}$ and ε is a very small positive value. In an example with k phases, the first $k-1$ phases consist of 5 intervals each, while the last phase has only 3 intervals. In the end, GREEDY_α accepts a single interval of weight $b^{2(k-1)}$, while the optimal solution contains two intervals of weight $b^{2i-1}-\varepsilon$, for $1 \leq i \leq k$, and one interval of weight $b^{2i}-\varepsilon$, for $1 \leq i \leq k-1$. It is easy to show that the ratio of GREEDY_α tends to the claimed bound as ε goes to zero and k goes to infinity. Again, the examples can be adapted to the case $m > 1$ by including m copies of every interval. □

This shows that the bound in Theorem 10 is tight for WJISP_1.

Corollary 12 *[equal lengths] For WJISP_1 with arbitrary weights, GREEDY_α performs best for $\alpha = 2\cos(\frac{2}{9}\pi) - 1 \approx 0.5321$, in which case it achieves approximation ratio*

$$\frac{1+2\cos(\frac{2}{9}\pi)}{7+6\cos(\frac{4}{9}\pi)-10\cos(\frac{2}{9}\pi)} \approx 6.638 \ .$$

Now we consider the case $m = 2$.

Theorem 13 *[equal lengths] For WJISP_2 with arbitrary weights, GREEDY_α achieves approximation ratio $\frac{2+\alpha}{\alpha(1-\alpha^2)}$.*

The proof of Theorem 13 is given in Appendix A. We obtain the following corollary.

Corollary 14 *[equal lengths] For WJISP_2 with arbitrary weights, GREEDY_α performs best for $\alpha = 2\cos(\frac{2}{9}\pi) - 1 \approx 0.5321$, in which case it achieves approximation ratio*

$$\frac{1+2\cos(\frac{2}{9}\pi)}{7+6\cos(\frac{4}{9}\pi)-10\cos(\frac{2}{9}\pi)} \approx 6.638 \ .$$

Now we consider the case of arbitrary $m > 1$.

Theorem 15 *[equal lengths] For $m > 1$, for WJISP_m with arbitrary weights, the approximation ratio of GREEDY_α is at most $\frac{2m-\alpha}{m\alpha(1-\alpha)}$.*

Proof: Charge the weight of intervals in OPT to intervals in T as in the proof of Theorem 5. We know that an interval $j \in T$ receives charge at most $\frac{2}{\alpha}w(j)$. Assume that an interval $j \in T \setminus A$ receives a charge of more than

$$\left(\frac{m-1}{m}+1\right)\frac{1}{\alpha}w(j) \ .$$

This implies that j receives charge from m intervals overlapping the right endpoint of j (each of these charges at most $\frac{1}{\alpha m}w(j)$) and from an interval $i \in OPT$ that belongs to the same job as j, that lies strictly to the right of j, and that is disjoint from j. As $j \in T \setminus A$ and j was in S when GREEDY$_\alpha$ processed interval i, this means that GREEDY$_\alpha$ selected an interval k that was processed after i and that belongs to the same job as j. Note that we must have $w(k) > w(j)/\alpha$. Distributing the total charge received by j and k proportionally among these two intervals, we can bound the resulting charge c_j and c_k for interval j and interval k, respectively, as in the proof of Theorem 10:

$$\begin{aligned} c_j &\leq w(j)\left(\frac{1}{\alpha}+\frac{1}{\alpha+1}\right) \\ c_k &\leq w(k)\left(\frac{1}{\alpha}+\frac{1}{\alpha+1}\right) \end{aligned}$$

So every interval $j \in T \setminus A$ that is not involved in the redistribution of charge receives charge at most $\left(\frac{m-1}{m}+1\right)\frac{1}{\alpha}w(j)$, while an interval $j \in T \setminus A$ that is involved in redistribution receives charge at most $w(j)\left(\frac{1}{\alpha}+\frac{1}{\alpha+1}\right)$. For $m \geq 2$ the former bound is larger, so that we can bound $w(OPT)$ as follows:

$$\begin{aligned} w(OPT) &\leq w(A)\cdot\frac{2}{\alpha}+w(T\setminus A)\frac{2m-1}{m\alpha} = w(A)\cdot\frac{1}{m\alpha}+w(T)\frac{2m-1}{m\alpha} \\ &\leq w(A)\cdot\frac{1}{m\alpha}+w(A)\frac{2m-1}{m\alpha(1-\alpha)} = w(A)\cdot\frac{2m-\alpha}{m\alpha(1-\alpha)} \end{aligned}$$

□

We do not now whether our analysis of Theorem 15 is tight. Nevertheless, we can determine the value of α that optimizes the obtained bound and get the following corollary.

Corollary 16 *[equal lengths] For WJISP$_m$, $m \geq 3$, with arbitrary weights, our analysis of GREEDY$_\alpha$ gives the best ratio for $\alpha = 2m - \sqrt{4m^2-2m}$, in which case the bound on the approximation ratio of GREEDY$_\alpha$ is*

$$\frac{4m-1+2\sqrt{4m^2-2m}}{m} \ .$$

Some bounds on the ratio of GREEDY$_\alpha$ resulting from Corollary 12, Corollary 14, and Corollary 16 are as follows:

m	1	2	3	4	5	6	7	8	9	10
ratio	6.638	6.638	7.318	7.491	7.594	7.663	7.712	7.748	7.776	7.799

Note that for every $m \geq 1$ these bounds are better than the ratio 8 that GREEDY$_\alpha$ achieves for WJISP$_m$ with arbitrary weights and arbitrary lengths. Furthermore, our bounds for the equal-lengths case converge to 8 as m goes to infinity.

4.2.2 Equal weights per job

The analysis for the case of WJISP$_m$ with equal lengths and equal weights per job is similar to the previous section. First, we analyze GREEDY$_\alpha$ for WJISP$_m$ with equal lengths and equal weights per job for the special case $m = 1$.

Theorem 17 *[equal lengths] For WJISP$_1$ with equal weights per job, GREEDY$_\alpha$ achieves approximation ratio* $\frac{1+\alpha(1-\alpha)}{\alpha(1-\alpha)}$.

Proof: Consider an interval $i \in OPT$ and the instant when GREEDY$_\alpha$ processed that interval. If $i \in T$, charge $w(i)$ to i and call this charge *overlap charge*. Assume that $i \notin T$. If S contained an interval k belonging to the same job as i and disjoint from i, charge $w(i)$ to k and call this charge *job charge*. Note that $k \in A$ in this case, as the intervals are processed in order of non-decreasing right endpoint and all intervals have the same length. Otherwise, S contained an interval j intersecting i and with weight $w(j) > \alpha w(i)$. Charge $w(i)$ to j and call this charge *overlap charge*.

The key observation is that an interval $k \in T$ can receive job charge only if $k \in A$. Furthermore, the job charge received by k is at most $w(k)$. Therefore, the total job charge is bounded by $w(A)$. The total overlap charge is obviously bounded by $\frac{1}{\alpha}w(T)$. Using $w(T) \leq w(A)/(1-\alpha)$, we obtain $w(OPT) \leq w(A) + \frac{1}{\alpha}w(T) \leq w(A)(1 + \frac{1}{\alpha(1-\alpha)})$. This shows $w(OPT) \leq \frac{1+\alpha(1-\alpha)}{\alpha(1-\alpha)}w(A)$. □

This result is tight as witnessed by the following theorem:

Theorem 18 *[equal lengths] For every $m \geq 1$, the approximation ratio of GREEDY$_\alpha$ for WJISP$_m$ with equal weights per job is not better than* $\frac{1+\alpha(1-\alpha)}{\alpha(1-\alpha)}$.

Proof: Consider the instance of WJISP$_1$ whose construction is sketched in Figure 8, where $b = \frac{1}{\alpha}$ and ε is a very small positive value. In an instance with k phases, GREEDY$_\alpha$ selects the intervals of weight 1, b, b^2, ..., b^{k-1} in turn, ending up with a single interval of weight b^{k-1} (drawn dashed). The optimal solution (drawn bold) contains the intervals with weight $b^i - \varepsilon$, $1 \leq i \leq k$, and one interval with weight b^{k-1}. Straightforward calculations show that the ratio between the optimal solution and the solution of GREEDY$_\alpha$ tends to $\frac{1+\alpha(1-\alpha)}{\alpha(1-\alpha)}$ as k goes to infinity and ε goes to zero. Furthermore, the same ratio can be achieved for $m > 1$ by creating m copies of every interval. □

This shows that Theorem 17 is tight.

Corollary 19 *[equal lengths] For WJISP$_1$ with equal weights per job, GREEDY$_\alpha$ performs best for $\alpha = \frac{1}{2}$, in which case it achieves approximation ratio 5.*

We will prove in Theorem 23 that no deterministic myopic algorithm for WJISP$_1$ with equal weights per job and arbitrary lengths can have approximation ratio better than approximately 5.828. This shows that the case of WJISP$_1$ with equal lengths is provably easier to approximate for myopic algorithms than the case of arbitrary lengths.

Now we consider the case $m > 1$.

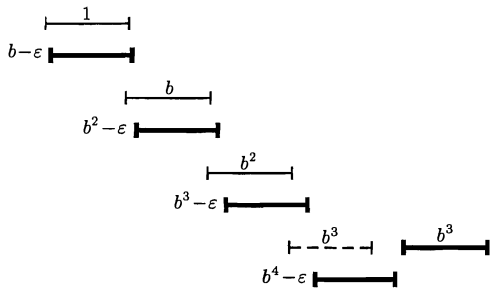


Figure 8: Worst-case example for equal lengths and equal weights per job

Theorem 20 *[equal lengths] For $m > 1$, for $WJISP_m$ with equal weights per job, the approximation ratio of $GREEDY_\alpha$ is not worse than*

$$\frac{1+\alpha(1-\alpha)}{\alpha(1-\alpha)} \text{ for } \alpha \leq \frac{1}{m} \quad \text{and} \quad \frac{m\alpha+m-\alpha}{m\alpha(1-\alpha)} \text{ for } \alpha \geq \frac{1}{m}.$$

Proof: Charge the weight of intervals in OPT to intervals in T as in the proof of Theorem 8. We know that an interval $j \in T$ receives charge at most $(\frac{1}{\alpha}+1)w(j)$. Assume that an interval $j \in T$ receives a charge of more than

$$w(j) \cdot \max\left\{\frac{1}{\alpha}, 1+\frac{m-1}{m\alpha}\right\} .$$

We claim that this implies that j receives charge from an interval $i \in OPT$ that belongs to the same job, is disjoint from j, and lies strictly to the right of j (which in turn implies that $j \in A$). To see this, assume that j does not receive charge from such an interval i. Then either j receives charge only from overlapping intervals belonging to different jobs (and the total charge of j is at most $m \cdot \frac{1}{m\alpha}w(j)$) or from one overlapping interval belonging to the same job (giving charge exactly $w(j)$) and at most $m-1$ overlapping intervals belonging to different jobs (giving total charge at most $\frac{m-1}{m\alpha}w(j)$).

So we can bound OPT as follows:

$$w(OPT) \leq w(A) \cdot (1+\frac{1}{\alpha}) + w(T \setminus A) \max\left\{\frac{1}{\alpha}, 1+\frac{m-1}{m\alpha}\right\}$$

Assume first that $\alpha \geq \frac{1}{m}$. Then $\frac{1}{\alpha} \leq 1+\frac{m-1}{m\alpha}$. In this case, we get:

$$\begin{aligned} w(OPT) &\leq w(A) \cdot (1+\frac{1}{\alpha}) + w(T \setminus A)(1+\frac{m-1}{m\alpha}) \\ &= w(A)\frac{1}{m\alpha} + w(T)\frac{m\alpha+m-1}{m\alpha} \\ &\leq w(A)\frac{m\alpha+m-\alpha}{m\alpha(1-\alpha)} \end{aligned}$$

Now assume that $\alpha \leq \frac{1}{m}$. In this case we have $\frac{1}{\alpha} \geq 1+\frac{m-1}{m\alpha}$. We get:

$$w(OPT) \leq w(A) \cdot (1+\frac{1}{\alpha}) + w(T \setminus A)\frac{1}{\alpha}$$

$$
\begin{aligned}
&= w(A) + w(T)\frac{1}{\alpha} \\
&\leq w(A)(1 + \frac{1}{\alpha(1-\alpha)})
\end{aligned}
$$

□

We do not know whether our analysis of Theorem 20 is tight. Nevertheless, we can determine the value of α that optimizes the obtained bound and get the following corollary.

Corollary 21 *[equal lengths] For $WJISP_m$ with equal weights per job, the following bounds for the ratio of $GREEDY_\alpha$ are the best bounds obtainable from our analysis:*

- $m = 2$*: ratio* 5*, obtained for* $\alpha = \frac{1}{2}$ *(derived from Theorem 20).*
- $3 \leq m \leq 67$*: ratio*

$$\frac{5m^3 - 3m^2 - 6m + 2 + \sqrt{5m^2 - 4m}(3m^2 + m - 2)}{2m(m^2 - 1)}$$

  for $\alpha = \frac{3m - \sqrt{5m^2 - 4m}}{2m+2}$ *(derived from Theorem 20).*

- $m \geq 68$*: ratio* $3 + 2\sqrt{2}$ *for* $\alpha = \sqrt{2} - 1$ *(derived from Corollary 9).*

Some bounds on the ratio of GREEDY_α resulting from Corollaries 19 and 21 are as follows:

m	1	2	3	4	5	6	7	8	9	10
ratio	5	5	5.268	5.417	5.505	5.564	5.606	5.637	5.661	5.681

Note that our bounds for the equal-lengths case are better than the ratio $3 + 2\sqrt{2} \approx 5.828$ that GREEDY_α achieves for WJISP_m with arbitrary lengths and equal weights per job for $1 \leq m \leq 67$.

5 Competitive lower bounds

In this section, we are interested in lower bounds on the best approximation ratio that can be achieved by any myopic algorithm for WJISP, and we use competitive analysis [5] to answer this question. We phrase our arguments in terms of an adversary who constructs worst-case instances for a myopic algorithm incrementally, depending on previous decisions made by the algorithm. In our illustrations, intervals belonging to the same job are always drawn in the same row. In Section 5.1 we deal with the unweighted case, Section 5.2 treats the case of equal weights per job, and Section 5.3 investigates the case of arbitrary weights. Section 5.4 gives a lower bound for randomized myopic algorithms for JISP.

5.1 The unweighted version of WJISP (JISP)

In the case of equal weights, it is easy to show that no myopic algorithm for WJISP_m can achieve approximation ratio better than 2.

Theorem 22 *No myopic algorithm for $JISP_m$ can achieve an approximation ratio better than* 2*, even if all intervals have equal length.*

Proof: Initially, the adversary presents a set Q of $2m$ intervals that belong to different jobs and that have the same right endpoint p. Let S be the set of at most m intervals that are selected by the algorithm at this time. Let S' be a subset of Q that contains all intervals in S and that has cardinality m. For every interval in S', the adversary presents an interval that belongs to the same job and that is to the right of p. All these new intervals have the same right endpoint. The optimal solution contains $2m$ intervals, while the algorithm accepts at most m intervals. Notice that the lengths of all intervals in this construction can be set equal. □

5.2 WJISP$_1$ with equal weights per job

Consider WJISP$_1$ with equal weights per job. In the following, we prove tight bounds on the best approximation ratio that can be achieved by any (deterministic) myopic algorithm for the case of arbitrary lengths or equal lengths per job and for the case of equal lengths. We view the construction of worst-case examples for a myopic algorithm as a game played by the algorithm with an adversary. In a move, the adversary presents a new interval, and in response the algorithm accepts this interval or rejects.

We begin with the case of equal lengths per job. For every $\varepsilon > 0$ that is sufficiently small (e.g., take $\frac{1}{8} \geq \varepsilon > 0$), the adversary has a strategy that forces the algorithm to create a solution of weight $(3 + 2\sqrt{2})/(1 + \varepsilon)^2$ times smaller than the optimum.

The strategy of the adversary has the following properties (some of them hold after the first two moves).

- The right endpoints of intervals presented by the adversary are strictly increasing.
- The tentative solution of the algorithm consists of exactly one interval called c (the current one).
- A set of intervals, P, forms a part of the eventual solution of the adversary, all elements of P have right endpoint smaller than the right endpoint of c and none belongs to the same job as c.
- Whenever the algorithm accepts a new interval, the ratio $w(P)/w(c)$ increases at least by a $1 + \varepsilon$ factor.
- If the algorithm rejects intervals sufficiently many times, the adversary wins by exhibiting a solution R such that $(1 + \varepsilon)^2 w(R)/w(c) \geq 3 + 2\sqrt{2}$.

The strategy of the adversary can be described as a set of prescriptions how to move in different states.

State 0: initial. In the first move the adversary presents interval i_0 with weight 1, and the algorithm has to accept i_0. In the second move, the adversary presents interval i_1 that belongs to a new job, intersects i_0 and has a larger right endpoint; $w(i_1) = 3 + 2\sqrt{2}$. If the algorithm does not accept i_1, the adversary exhibits $R = \{i_1\}$ and wins. Otherwise, we have $c = i_1$ and $P = \{i_0\}$.

The remaining states depend on the algorithm response in the previous move. Intervals of the form a_i belong to new jobs, have weight $w(a_i) = w(c)(1 + \varepsilon)^i$, and every a_i has the

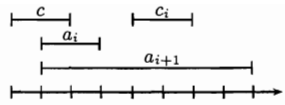


Figure 9: Illustration of intervals a_i, c_i, and a_{i+1} played by the adversary.

same left endpoint such that a_i intersects c but none of the intervals in P. An interval of the form c_i belongs to the same job as c (thus it has the same weight and length) and is to the right of a_i (so that it does not intersect a_i). See Figure 9 for an illustration.

State 1: the algorithm accepted a new interval. The adversary presents an interval of a new job, a_{-1}. If the algorithm accepts a_{-1}, then c becomes a_{-1}, P remains unchanged and $w(P)/w(c)$ increases by the factor of $w(c)/w(a_{-1}) = 1+\varepsilon$.

State 2: the algorithm rejected a_i. The adversary presents c_i. Even if the algorithm accepts c_i, its solution still consists of only one interval, of the same weight as before. The adversary checks $R_i = P \cup \{a_i, c_i\}$. If $(1+\varepsilon)^2 w(R_i)/w(c) \geq 3+2\sqrt{2}$, it wins. Otherwise, one can see that $w(P)/w(c) < 4$. If the algorithm accepts c_i, the adversary inserts a_i into P and the ratio $w(P)/w(c)$ increases by a factor larger than $6/5$.

State 3: the algorithm rejected c_i. The adversary presents a_{i+1}. If the algorithm accepts a_{i+1}, the adversary replaces P with R_i.

The adversary wins in State 2 under the condition that $w(R_i) = w(P)+w(c)+w(c)(1+\varepsilon)^i$ is sufficiently large in comparison with $w(c)$, or, more precisely, if

$$(1+\varepsilon)^2 \frac{w(P)+w(c)(1+\varepsilon)^i+w(c)}{w(c)} \geq 3+2\sqrt{2}.$$

We can note that as i grows, the left-hand-side increases and eventually the adversary has to win if the algorithm keeps rejecting intervals a_i. Let $y = w(P)/w(c)$ and $z = (1+\varepsilon)^i$. If the adversary did not win, we have $(1+\varepsilon)^2(y+z+1) < 3+2\sqrt{2}$.

Now suppose that the algorithm accepted a_{i+1} in State 3. Then the ratio $w(P)/w(c)$ increases by

$$\frac{w(R_i)}{w(c)(1+\varepsilon)^{i+1}} \cdot \frac{w(c)}{w(P)} = \frac{y+z+1}{(1+\varepsilon)zy} \geq \frac{4(y+z+1)}{(1+\varepsilon)(y+z)^2} = \frac{4(x+1)}{(1+\varepsilon)x^2}$$

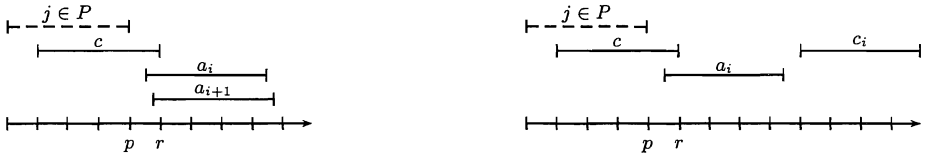
where $x = y+z$. Note that the last expression is a decreasing function of x. Because the adversary did not win when the algorithm was rejecting c_i, we had $(1+\varepsilon)^2(x+1) < 3+2\sqrt{2}$ and thus our estimate for the increase of $w(P)/w(c)$ is the smallest for $x+1 = (3+2\sqrt{2})(1+\varepsilon)^{-2}$:

$$\frac{4(x+1)}{(1+\varepsilon)x^2} \geq \frac{4(3+2\sqrt{2})(1+\varepsilon)^{-2}}{(1+\varepsilon)((3+2\sqrt{2})(1+\varepsilon)^{-2}-1)^2} =$$

$$\frac{4(3+2\sqrt{2})(1+\varepsilon)}{(3+2\sqrt{2}-(1+\varepsilon)^2)^2} = \frac{4(3+2\sqrt{2})(1+\varepsilon)}{(2+2\sqrt{2}-2\varepsilon-\varepsilon^2)^2} >$$

$$\frac{4(3+2\sqrt{2})(1+\varepsilon)}{(2+2\sqrt{2})^2} = 1+\varepsilon.$$

As we can choose ε arbitrarily small, we obtain the following theorem.



(a) regular step (b) final step

Figure 10: Illustration of the intervals used by the adversary in the case of equal lengths.

Theorem 23 *No myopic algorithm for WJISP$_1$ with equal weights per job can achieve approximation ratio better than $3 + 2\sqrt{2} \approx 5.828$. This lower bound applies in the case of arbitrary lengths and in the case of equal lengths per job.*

Now consider WJISP$_1$ with equal weights per job and equal lengths. We can prove that no myopic algorithm can achieve approximation ratio better than 5. In particular, we will see that the adversary can force the algorithm to create a solution whose weight is smaller than the optimum by a factor of $5/(1+\varepsilon)^2$ for any $\frac{1}{8} \geq \varepsilon > 0$.

The adversary's strategy and its analysis is analogous to the one we used to obtain Theorem 23. The only difference is that an interval of type c_i is played only at the end of the game, as illustrated in Figure 10. More precisely, this means that we replace the original description of State 2 by the following.

State 2′: the algorithm rejected a_i. The adversary checks whether $(w(P) + w(a_i) + w(c_i))/w(c) \geq 5/(1+\varepsilon)^2$. If this is true, the adversary presents c_i. No matter whether the algorithm accepts c_i or not, its solution still consists of only one interval, of the same weight as before. Therefore, the adversary wins by exhibiting $R_i = P \cup \{a_i, c_i\}$ in this case. Otherwise, the adversary does not present c_i. Instead, it presents a_{i+1} and, if the algorithm accepts a_{i+1}, c becomes a_{i+1} and the adversary replaces P with $P \cup \{a_i\}$.

Note that State 3 is no longer needed in this adversary strategy. The adversary wins in State 2′ under the condition that $w(R_i) = w(P) + w(c) + w(c)(1+\varepsilon)^i$ is sufficiently large in comparison with $w(c)$, or, more precisely,

$$(1+\varepsilon)^2 \frac{w(P) + w(c)(1+\varepsilon)^i + w(c)}{w(c)} \geq 5.$$

We can note that as i grows, the left-hand-side increases and eventually the adversary has to win if the algorithm keeps rejecting intervals a_i. Let $y = w(P)/w(c)$ and $z = (1+\varepsilon)^i$. If the adversary did not win, we have $(1+\varepsilon)^2(y+z+1) < 5$.

Now suppose that the algorithm accepted a_{i+1} in State 2′. Then the ratio $w(P)/w(c)$ increases by

$$\frac{w(P \cup \{a_i\})}{w(c)(1+\varepsilon)^{i+1}} \cdot \frac{w(c)}{w(P)} = \frac{y+z}{(1+\varepsilon)zy} \geq \frac{4(y+z)}{(1+\varepsilon)(y+z)^2} = \frac{4}{(1+\varepsilon)x}$$

where $x = y + z$. Note that the last expression is a decreasing function of x. Because the adversary would not have won by presenting c_i, we had $(1+\varepsilon)^2(x+1) < 5$ and thus our estimate for the increase of $w(P)/w(c)$ is the smallest for $x = 5(1+\varepsilon)^{-2} - 1$:

$$\frac{4}{(1+\varepsilon)x} \geq \frac{4}{(1+\varepsilon)(5(1+\varepsilon)^{-2} - 1)} =$$

$$\frac{4(1+\varepsilon)}{5-(1+\varepsilon)^2} \geq \frac{4(1+\varepsilon)}{4} = 1+\varepsilon.$$

As we can choose ε arbitrarily small, we obtain the following theorem.

Theorem 24 *No myopic algorithm for WJISP$_1$ with equal weights per job and equal lengths can achieve approximation ratio better than* 5.

Corollary 19 showed that the myopic algorithm GREEDY$_\alpha$ with $\alpha = \frac{1}{2}$ achieves approximation ratio 5 for WJISP$_1$ with equal weights per job and equal lengths. Thus, Theorem 24 implies that GREEDY$_\alpha$ is optimal within the class of myopic algorithms in this case. Note that GREEDY$_\alpha$ decides whether it accepts a new interval without taking into account the amount of overlap between a currently accepted interval and the new interval. Intuitively, one might think that considering the amount of overlap would give an advantage to a myopic algorithm, but our tight lower bound shows that this is not true in the worst case.

5.3 WJISP$_1$ with arbitrary weights

To obtain lower bounds in the case of WJISP$_1$ with arbitrary weights, we can employ essentially the same adversary strategy as in the previous section. The only difference is that the interval c_i, which belongs to the same job as c, can now have a weight different from $w(c)$.

First, let us define q as the minimum of the function

$$f(x) = \frac{x^2+x}{x-\frac{1}{x}-1}$$

in the range $((1+\sqrt{5})/2, \infty)$. This minimum is attained at

$$x = \hat{x} = \frac{2}{3} + \sqrt[3]{\frac{71}{27} + \sqrt{\frac{35}{27}}} + \frac{16}{9\sqrt[3]{\frac{71}{27} + \sqrt{\frac{35}{27}}}} \approx 3.365$$

with $q = f(\hat{x}) \geq 7.103$. We will show that the adversary can force the algorithm to create a solution whose weight is smaller than the optimum by a factor of $q/(1+\varepsilon)^2$ for any $1/40 \geq \varepsilon > 0$. The adversary strategy is based on States 0–3 of the previous section, with the following modifications for State 0 and State 2:

State 0″: initial. In the first move the adversary presents interval i_0 with weight 1, and the algorithm has to accept i_0. In the second move, the adversary presents interval i_1 that belongs to a new job, intersects i_0 and has a larger right endpoint; $w(i_1) = 8$. If the algorithm does not accept i_1, the adversary exhibits $R = \{i_1\}$ and wins. Otherwise, we have $c = i_1$ and $P = \{i_0\}$.

State 2″: the algorithm rejected a_i. The adversary presents c_i with weight $w(c_i) = w(c)(1+w(a_i)/w(P))/(1+\varepsilon)$. If the algorithm accepts c_i, then c becomes c_i and the adversary inserts a_i into P. The ratio increases by

$$\frac{w(P)+w(a_i)}{w(c_i)} \cdot \frac{w(c)}{w(P)} = \frac{w(P)+w(a_i)}{w(c)(1+w(a_i)/w(P))/(1+\varepsilon)} \cdot \frac{w(c)}{w(P)} =$$

$$\frac{(1+\varepsilon)(w(P)+w(a_i))}{w(P)+w(a_i)} = 1+\varepsilon,$$

where the variable P refers to the status of P before inserting a_i. If the algorithm rejects c_i, the adversary checks whether $R_i = P \cup \{a_i, c_i\}$ satisfies $(1+\varepsilon)^2 w(R_i)/w(c) \geq q$ and, if so, wins by exhibiting R_i.

The adversary wins in State 2'' if the algorithm rejects c_i and if $w(R_i) = w(P) + w(a_i) + w(c)(1 + w(a_i)/w(P))/(1+\varepsilon)$ is sufficiently large in comparison with $w(c)$ or, more precisely,

$$(1+\varepsilon)^2 \frac{w(P) + w(c)(1+\varepsilon)^i + w(c)(1+w(a_i)/w(P))/(1+\varepsilon)}{w(c)} \geq q.$$

We can note that as i grows, the left-hand-side increases and eventually the adversary has to win if the algorithm keeps rejecting intervals a_i and c_i. Let $y = w(P)/w(c)$ and $z = (1+\varepsilon)^i$. If the adversary did not win, we have

$$(1+\varepsilon)^2 \left(y + z + \frac{1}{1+\varepsilon} + \frac{z}{(1+\varepsilon)y} \right) = (1+\varepsilon)^2 (y+z) \left(1 + \frac{1}{(1+\varepsilon)y} \right) < q. \qquad (4)$$

Now suppose that the algorithm accepted a_{i+1} in State 3. Then the ratio $w(P)/w(c)$ increases by

$$\frac{w(R_i)}{w(c)(1+\varepsilon)^{i+1}} \cdot \frac{w(c)}{w(P)} = (y+z)\frac{1+\frac{1}{(1+\varepsilon)y}}{(1+\varepsilon)zy} =$$

$$\left(\frac{1}{z} + \frac{1}{y}\right) \cdot \left(\frac{(1+\varepsilon)y+1}{(1+\varepsilon)^2 y}\right)$$

This is a decreasing function of z. From (4) we get that $z < \frac{q}{(1+\varepsilon)^2(1+\frac{1}{y(1+\varepsilon)})} - y$. (Since $z > 0$, this means that we must have $q > y(1+\varepsilon)^2(1+\frac{1}{y(1+\varepsilon)}) = (1+\varepsilon)(y(1+\varepsilon)+1)$.) So our bound on the increase of $w(P)/w(c)$ is at least

$$\left(\frac{1}{\frac{q}{(1+\varepsilon)^2(1+\frac{1}{y(1+\varepsilon)})} - y} + \frac{1}{y} \right) \cdot \frac{(1+\varepsilon)y+1}{(1+\varepsilon)^2 y} =$$

$$\left(\frac{(1+\varepsilon)^2(1+\frac{1}{y(1+\varepsilon)})}{q - y(1+\varepsilon)^2(1+\frac{1}{y(1+\varepsilon)})} + \frac{1}{y} \right) \cdot \frac{(1+\varepsilon)y+1}{(1+\varepsilon)^2 y} =$$

$$\frac{y(1+\varepsilon)^2(1+\frac{1}{y(1+\varepsilon)}) + q - y(1+\varepsilon)^2(1+\frac{1}{y(1+\varepsilon)})}{yq - y^2(1+\varepsilon)^2(1+\frac{1}{y(1+\varepsilon)})} \cdot \frac{(1+\varepsilon)y+1}{(1+\varepsilon)^2 y} =$$

$$\frac{q}{q - y(1+\varepsilon)^2(1+\frac{1}{y(1+\varepsilon)})} \cdot \frac{(1+\varepsilon)y+1}{(1+\varepsilon)^2 y^2} =$$

$$\frac{q}{q - (1+\varepsilon)(y(1+\varepsilon)+1)} \cdot \frac{(1+\varepsilon)y+1}{(1+\varepsilon)^2 y^2}$$

With $x = y(1+\varepsilon)$ this becomes:

$$\frac{q}{q - (1+\varepsilon)(x+1)} \cdot \frac{x+1}{x^2}$$

and we want to show

$$\frac{q}{q-(1+\varepsilon)(x+1)} \cdot \frac{x+1}{x^2} \geq 1+\varepsilon.$$

This is equivalent to

$$q \cdot \frac{x+1}{x^2} \geq (1+\varepsilon)(q-(1+\varepsilon)(x+1))$$

$$\Leftrightarrow q(1+\varepsilon-\frac{x+1}{x^2}) \leq (1+\varepsilon)^2(x+1)$$

If $1+\varepsilon-\frac{x+1}{x^2} < 0$, this holds. Otherwise, we continue our calculation as follows.

$$q \leq \frac{(1+\varepsilon)^2(x+1)}{1+\varepsilon-\frac{x+1}{x^2}}$$

$$\Leftrightarrow q \leq \frac{(1+\varepsilon)^2x^2(x+1)}{(1+\varepsilon)x^2-x-1}$$

If $x \geq 1+\sqrt{3}$, then we have $x^2-2x-2 \geq 0$ and thus $(1+\varepsilon)x^2-x-1 \leq (1+2\varepsilon)(x^2-x-1)$. Since q minimizes the function $f(x) = \frac{x^3+x^2}{x^2-x-1}$, we get $q \leq \frac{x^3+x^2}{x^2-x-1} \leq \frac{(1+\varepsilon)^2x^2(x+1)}{(1+2\varepsilon)(x^2-x-1)} \leq \frac{(1+\varepsilon)^2x^2(x+1)}{(1+\varepsilon)x^2-x-1}$, as desired. If $x \leq 1+\sqrt{3}$, the bound $\frac{(1+\varepsilon)^2x^2(x+1)}{(1+\varepsilon)x^2-x-1}$ is minimized for $x = 1+\sqrt{3}$ (since the function is decreasing when x is between the larger root of $(1+\varepsilon)x^2-x-1$ and $1+\sqrt{3}$), so that it suffices to establish

$$q \leq \frac{(1+\varepsilon)^2(4+2\sqrt{3})(2+\sqrt{3})}{(1+\varepsilon)(4+2\sqrt{3})-2-\sqrt{3}}.$$

Since we have $\varepsilon \leq 1/40$, it suffices to show

$$q \leq \frac{(4+2\sqrt{3})(2+\sqrt{3})}{(41/40)(4+2\sqrt{3})-2-\sqrt{3}} \approx 7.109.$$

This holds, because we have $q \approx 7.103$. Hence, we have shown that the ratio $w(P)/w(c)$ increases at least by a factor of $1+\varepsilon$. We obtain the following theorem.

Theorem 25 *No myopic algorithm for WJISP$_1$ with arbitrary weights can achieve approximation ratio better than* 7.103. *This lower bound applies in the case of arbitrary lengths and in the case of equal lengths per job.*

Now consider WJISP$_1$ with arbitrary weights and equal lengths. We can prove that no myopic algorithm can achieve approximation ratio better than $3+2\sqrt{2}$. In particular, we will see that the adversary can force the algorithm to create a solution whose weight is smaller than the optimum by a factor of $(3+2\sqrt{2})/(1+\varepsilon)^2$ for any $\frac{1}{8} \geq \varepsilon > 0$.

The adversary's strategy and its analysis is analogous to the one we used to obtain Theorem 25. The only difference is that an interval of type c_i is played only at the end of the game, i.e., if $(w(P)+w(a_i)+w(c_i))/w(c)$ is large enough for the adversary to win. More precisely, this means that we replace the State 2″ by the following.

State 2‴: the algorithm rejected a_i. The adversary considers presenting c_i with weight $w(c_i) = w(c)(1+w(a_i)/w(P))/(1+\varepsilon)$. The adversary checks whether $(w(P)+w(a_i)+$

$w(c_i))/w(c) \geq (3+2\sqrt{2})/(1+\varepsilon)^2$. If this is true, the adversary presents c_i. If the algorithm accepts c_i, then c becomes c_i and the adversary inserts a_i into P. The ratio increases by

$$\frac{w(P)+w(a_i)}{w(c_i)} \cdot \frac{w(c)}{w(P)} = \frac{w(P)+w(a_i)}{w(c)(1+w(a_i)/w(P))/(1+\varepsilon)} \cdot \frac{w(c)}{w(P)} =$$

$$\frac{(1+\varepsilon)(w(P)+w(a_i))}{w(P)+w(a_i)} = 1+\varepsilon\,,$$

where the variable P refers to the status of P before inserting a_i. If the algorithm rejects c_i, the adversary wins by exhibiting $R_i = P \cup \{a_i, c_i\}$. If the condition $(w(P)+w(a_i)+w(c_i))/w(c) \geq (3+2\sqrt{2})/(1+\varepsilon)^2$ does not hold, the adversary does not present c_i. Instead, it presents a_{i+1} and, if the algorithm accepts a_{i+1}, c becomes a_{i+1} and the adversary replaces P with $P \cup \{a_i\}$.

Note that State 3 is no longer needed in this adversary strategy. If the adversary plays c_i in State 2‴, the outcome is always beneficial for the adversary: if the algorithm accepts c_i, the ratio $w(P)/w(c)$ increases by a factor $1+\varepsilon$, and if the algorithm rejects c_i, then the adversary wins immediately. Furthermore, it is clear that if the algorithm keeps rejecting intervals a_i, the condition for playing c_i, which is

$$(1+\varepsilon)^2 \frac{w(P)+w(c)(1+\varepsilon)^i + w(c_i)}{w(c)} \geq 3+2\sqrt{2},$$

will be fulfilled at some time (the left-hand side grows with i). Therefore, it remains to show that the adversary also benefits if the algorithm accepts a_{i+1}.

Let $y = w(P)/w(c)$ and $z = (1+\varepsilon)^i$. If the adversary did not play c_i, we have $(1+\varepsilon)^2(y+z)(1+1/(y(1+\varepsilon))) < 3+2\sqrt{2}$, and this implies

$$z < \frac{3+2\sqrt{2}}{(1+\varepsilon)^2(1+\frac{1}{y(1+\varepsilon)})} - y\ . \tag{5}$$

Now suppose that the algorithm accepted a_{i+1} in State 2‴. Then the ratio $w(P)/w(c)$ increases by

$$\frac{w(P \cup \{a_i\})}{w(c)(1+\varepsilon)^{i+1}} \cdot \frac{w(c)}{w(P)} = \frac{y+z}{(1+\varepsilon)zy} = \frac{1}{1+\varepsilon}\left(\frac{1}{y}+\frac{1}{z}\right).$$

Note that the last expression is a decreasing function of z. Using (5), we get that our estimate for the increase of $w(P)/w(c)$ is the smallest for $z = \frac{3+2\sqrt{2}}{(1+\varepsilon)^2(1+\frac{1}{y(1+\varepsilon)})} - y$. We can continue our calculation as follows.

$$\frac{1}{1+\varepsilon}\left(\frac{1}{y}+\frac{1}{z}\right) \geq \frac{1}{1+\varepsilon}\left(\frac{1}{y}+\frac{1}{\frac{3+2\sqrt{2}}{(1+\varepsilon)^2(1+\frac{1}{y(1+\varepsilon)})} - y}\right) \geq$$

$$\frac{1}{1+\varepsilon} \cdot \frac{3+2\sqrt{2}}{y(3+2\sqrt{2}-y(1+\varepsilon)^2(1+\frac{1}{y(1+\varepsilon)}))}$$

We want to show that this bound is at least $1+\varepsilon$. This is equivalent to showing $3+2\sqrt{2} \geq (1+\varepsilon)^2 y(3+2\sqrt{2}-y(1+\varepsilon)^2(1+\frac{1}{y(1+\varepsilon)}))$. Substituting x for $y(1+\varepsilon)^2$, this is equivalent to $3+2\sqrt{2} \geq x(2+2\sqrt{2}-\varepsilon-x)$. This inequality is obviously correct, since the function $f(x) =$

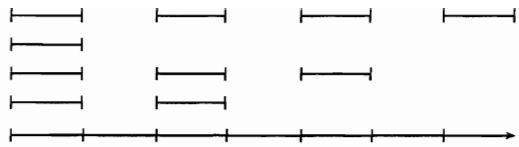


Figure 11: Example of JISP instance used in randomized lower bound.

$x(2+2\sqrt{2}-\varepsilon-x)$ is maximized at $\hat{x} = 1+\sqrt{2}-\varepsilon/2$ with $f(\hat{x}) = (1+\sqrt{2}-\varepsilon/2)^2 < 3+2\sqrt{2}$. Thus, we have shown that the ratio $w(P)/w(c)$ increases at least by a factor of $1+\varepsilon$.

As we can choose ε arbitrarily small, we obtain the following theorem.

Theorem 26 *No myopic algorithm for $WJISP_1$ with arbitrary weights and equal lengths can achieve approximation ratio better than $3+2\sqrt{2} \approx 5.828$.*

5.4 A randomized lower bound

In Section 5.1 we have shown that no (deterministic) myopic algorithm for JISP can have approximation ratio better than 2. Here we derive a bound on the best approximation ratio that can possibly be achieved by a *randomized myopic* algorithm against an oblivious adversary [5]. The randomized lower bound that we obtain is weaker than the deterministic bound, and it remains an open problem whether randomized myopic algorithms can in fact beat deterministic myopic algorithms for JISP. The following theorem shows that the use of randomization could at best improve the approximation ratio from 2 to approximately 1.582.

Theorem 27 *No randomized myopic algorithm for JISP can achieve an approximation ratio better than $\frac{e}{e-1} \approx 1.582$.*

Proof: We specify a probability distribution on instances with n jobs and show that the expected number of intervals accepted by a deterministic myopic algorithm is at most $\frac{e-1}{e}n+1$, while an optimal solution consists of n intervals. By Yao's principle (see e.g. Sections 2.2.2 and 13.3 of [20]), this implies the lower bound for randomized myopic algorithms against an oblivious adversary.

We consider a very restricted subset of instances of JISP, defined as follows. If such an instance consists of n jobs, then all its intervals have unit length and any two intervals either have identical endpoints or are disjoint. Initially, one interval with left endpoint 0 and right endpoint 1 is presented from each of the n jobs. Further intervals have left endpoints 2, 4, 6, ..., $2n-2$. We refer to the presentation of intervals with left endpoint $2i$ as *round i*. Let J_i denote the set of jobs from which intervals are presented in round i. The instances we construct always satisfy $J_{i+1} \subseteq J_i$ and $|J_{i+1}| = |J_i| - 1$. We say that the unique job in $J_i \setminus J_{i+1}$ *dies* after round i. An example of such an instance is shown in Figure 11. Note that an optimal solution to such an instance always contains n intervals.

The probability distribution is defined by choosing the job that dies after round i uniformly at random among the $n-i$ jobs from which an interval was presented in round i.

Consider any deterministic myopic algorithm A. If the set of intervals that are currently selected by A does not contain an interval from job j, we say that job j is *available* to A. We can assume without loss of generality that A always selects an interval in round i if at least one interval from an available job is presented in that round.

Let $E_{k,\ell}$ denote the expected number of intervals added to the solution by algorithm A until the end of the game provided that the current round consists of k intervals and ℓ of these k intervals belong to available jobs. Note that $E_{n,n}$ is just the expected number of intervals in the solution computed by A.

The following equations hold:

$$E_{k,0} = 0 \tag{6}$$

$$E_{k,1} = 1 \tag{7}$$

$$E_{k,\ell} = 1 + \frac{\ell-1}{k} E_{k-1,\ell-2} + \frac{k-\ell+1}{k} E_{k-1,\ell-1} \qquad \text{for } 1 < \ell \leq k \tag{8}$$

We need to explain why (8) holds. If $\ell > 0$, the algorithm always selects an interval in the current round. After that, among the k intervals of the current round there are $\ell-1$ remaining intervals belonging to available jobs. With probability $\frac{\ell-1}{k}$, one of these $\ell-1$ jobs dies, and with probability $\frac{k-\ell+1}{k}$, one of the other jobs dies.

We claim that (6)–(8) imply

$$E_{k,\ell} \leq k(1 - e^{-\ell/k}) + \frac{\ell}{k} \tag{9}$$

for all $0 \leq \ell \leq k$. For $k = \ell = n$, (9) becomes $E_{n,n} \leq \frac{e-1}{e} n + 1$. This shows that the expected number of intervals in the solution computed by A cannot be greater than γn for any $\gamma > \frac{e-1}{e}$, thus establishing the theorem.

We prove (9) by induction on ℓ. For $\ell = 0$, (9) is clearly true. For $\ell = 1$, the left-hand side of (9) is 1, and we must show that the right-hand side $k(1 - e^{-1/k}) + \frac{1}{k}$ is at least 1 for all $k \geq 1$. This can be proved in a straightforward way by considering the function $f(x) = x(1 - e^{-1/x}) + \frac{1}{x}$: this function satisfies $f(1) > 1$, $\lim_{x\to\infty} f(x) = 1$, and is monotonically decreasing for $0 < x < \infty$.

Now take $\ell > 1$ and assume that (9) holds for any value of k if ℓ is replaced by $\ell - 1$ or by $\ell - 2$ (inductive hypothesis). Starting with (8) and plugging in the inductive hypothesis for $E_{k-1,\ell-2}$ and $E_{k-1,\ell-1}$, we obtain:

$$\begin{aligned}
E_{k,\ell} &\leq 1 + \frac{\ell-1}{k}\left((k-1)(1 - e^{-(\ell-2)/(k-1)}) + \frac{\ell-2}{k-1}\right) \\
&\quad + \frac{k-\ell+1}{k}\left((k-1)(1 - e^{-(\ell-1)/(k-1)}) + \frac{\ell-1}{k-1}\right) \\
&= k + \frac{\ell-1}{k} - \frac{k-1}{k}\left((\ell-1)e^{-(\ell-2)/(k-1)} + (k-\ell+1)e^{-(\ell-1)/(k-1)}\right) \\
&= k + \frac{\ell-1}{k} - \frac{k-1}{k} e^{-\ell/k}\left((\ell-1)e^{\ell/k-(\ell-2)/(k-1)} + \right. \\
&\qquad \left. (k-\ell+1)e^{\ell/k-(\ell-1)/(k-1)}\right) \\
&\quad \ll \text{ now we use } e^x \geq 1 + x \gg \\
&\leq k + \frac{\ell-1}{k} - \frac{k-1}{k} e^{-\ell/k}\left((\ell-1)(1 + \frac{\ell}{k} - \frac{\ell-2}{k-1}) + \right. \\
&\qquad \left. (k-\ell+1)(1 + \frac{\ell}{k} - \frac{\ell-1}{k-1})\right) \\
&= k + \frac{\ell-1}{k} - \frac{k-1}{k} e^{-\ell/k}(k+1)
\end{aligned}$$

$$= k - ke^{-\ell/k} + \frac{\ell}{k} + \frac{e^{-\ell/k}}{k} - \frac{1}{k}$$
$$\leq k(1 - e^{-\ell/k}) + \frac{\ell}{k}$$

The last inequality follows from $e^{-\ell/k} \leq 1$. □

By considering only instances of JISP like the ones constructed in the proof of Theorem 27 we obtain a restricted version of JISP. An application of this restricted version could be as follows. A shop sells n different items. After each week, one of the items becomes unavailable, i.e., after i weeks only $n - i$ of the original items can be bought. Assume that you can afford to buy at most one item per week and that you would like to buy as many different items as possible. This problem is just the restricted version of JISP.

Note that no deterministic myopic algorithm can buy more than $n/2$ items in the worst case: every week, the adversary decides that one of the items that was not yet bought by the algorithm becomes unavailable. After $n/2$ weeks, no items are left to buy for the algorithm. So the deterministic lower bound of 2 from Theorem 22 applies even to this restricted version of JISP.

6 Conclusions

The weighted job interval selection problem has applications in diverse areas. We have studied several variants of this problem: we distinguished the case of arbitrary lengths and the case of equal lengths, and for each of these cases we investigated the setting with all weights equal, equal weights per job, and arbitrary weights. The case of equal lengths can be seen as a natural online scheduling problem. We showed that a simple algorithm called GREEDY$_\alpha$, which belongs to the class of so-called myopic algorithms, outputs a solution with a value that is within a constant factor of the value of an optimal solution. Together with our lower bound results, this implies that for the case of arbitrary lengths no myopic algorithm can do better than GREEDY$_\alpha$ (with respect to approximation ratio) in the unweighted case (for all m) and in the case of equal weights per job (for $m = 1$). In case of arbitrary weights a (small) gap remains. For the case of equal lengths, our results show that GREEDY$_\alpha$ is best possible among the myopic algorithms in case of equal weights per job (for $m = 1$).

An interesting question for future research is whether the use of randomization in myopic algorithms for WJISP can lead to improved approximation ratios. For the case of $JISP_1$, we showed that randomization could at best improve the approximation ratio from 2 to $e/(e-1) \approx 1.582$. For the weighted versions of WJISP, we do not have any strong randomized lower bounds. However, we can remark that all our lower bounds for deterministic myopic algorithms hold also for randomized algorithms against an *adaptive* adversary (i.e., an adversary that can react to the random choices of the algorithm).

Acknowledgments

We are deeply indebted to two anonymous referees for *Journal of Algorithms* who provided numerous comments and suggestions that helped to improve the paper. In particular, the first referee proposed the use of a union-find data structure to speed up our algorithm for

computing the cheapest conflict set (Theorem 1) and found a substantial simplification for the proofs of our lower bound results in Sections 5.2 and 5.3.

Furthermore, we would like to thank Eric Torng for bringing the caching application to our attention. We would also like to thank Baruch Schieber and Seffi Naor for discussions regarding the relationship between WJISP and TCSP. Finally, we thank Piotr Berman for supplying preliminary versions of the papers [4] and [27].

References

[1] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi. *Complexity and Approximation. Combinatorial Optimization Problems and their Approximability Properties.* Springer, Berlin, 1999.

[2] A. Bar-Noy, R. Bar-Yehuda, A. Freund, J. S. Naor, and B. Schieber. A unified approach to approximating resource allocation and scheduling. *Journal of the ACM*, 48(5):1069–1090, 2001.

[3] A. Bar-Noy, S. Guha, J. S. Naor, and B. Schieber. Approximating the throughput of multiple machines in real-time scheduling. *SIAM Journal on Computing*, 31(2):331–352, 2001.

[4] P. Berman and B. DasGupta. Multi-phase algorithms for throughput maximization for real-time scheduling. *Journal of Combinatorial Optimization*, 4:307–323, 2000.

[5] A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis.* Cambridge University Press, 1998.

[6] R. Canetti and S. Irani. Bounding the power of preemption in randomized scheduling. *SIAM Journal on Computing*, 27(4):993–1015, 1998.

[7] M. C. Carlisle and E. L. Lloyd. On the k-coloring of intervals. *Discrete Applied Mathematics*, 59:225–235, 1995.

[8] J. Chuzhoy, R. Ostrovsky, and Y. Rabani. Approximation algorithms for the job interval selection problem and related scheduling problems. In *Proceedings of the 42nd Annual Symposium on Foundations of Compute Science FOCS'01*, 2001.

[9] Y. Crama, O. Flippo, J. van de Klundert, and F. Spieksma. The assembly of printed circuit boards: a case with multiple machines and multiple board types. *European Journal of Operational Research*, 98:457–472, 1997.

[10] T. Erlebach and F. Spieksma. Simple algorithms for a weighted interval selection problem. In *Proceedings of the 11th Annual International Symposium on Algorithms and Computation ISAAC'00*, LNCS 1969, pages 228–240, 2000.

[11] M. Fischetti, S. Martello, and P. Toth. Approximation algorithms for fixed job schedule problems. *Operations Research*, 40:S96–S108, 1992.

[12] A. Frank. Some polynomial algorithms for certain graphs and hypergraphs. In *Proceedings of the 5th British Combinatorial Conference*, pages 211–226, 1975.

[13] M. L. Fredman and D. E. Willard. Trans-dichotomous algorithms for minimum spanning trees and shortest paths. *Journal of Computer and System Sciences*, 48(3):533–551, 1994.

[14] H. N. Gabow and R. E. Tarjan. A linear-time algorithm for a special case of disjoint set union. *Journal of Computer and System Sciences*, 30:209–221, 1985.

[15] S. A. Goldman, J. Parwatikar, and S. Suri. Online scheduling with hard deadlines. *Journal of Algorithms*, 34(2):370–389, 2000.

[16] M. H. Goldwasser. Patience is a virtue: The effect of slack on competitiveness for admission control. In *Proceedings of the 10th Annual ACM–SIAM Symposium on Discrete Algorithms SODA '99*, pages 396–405, 1999.

[17] D. Hochbaum. *Approximation algorithms for NP-hard problems.* PWS Publishing Company, Boston, 1997.

[18] S. v. Hoesel and R. Müller. Optimization in electronic markets: Examples in combinatorial auctions. *Netnomics*, 3(1):23–33, 2001.

[19] L. Kroon, M. Salomon, and L. van Wassenhove. Exact and approximation algorithms for the tactical fixed interval scheduling problem. *Operations Research*, 45:624–638, 1997.

[20] R. Motwani and P. Raghavan. *Randomized Algorithms.* Cambridge University Press, 1995.

[21] R. Raman. Priority queues: Small, monotone and trans-dichotomous. In *Proceedings of the 4th Annual European Symposium on Algorithms ESA '96*, LNCS 1136, pages 121–137, 1996.

[22] M. H. Rothkopf, A. Pekeč, and R. M. Harstad. Computationally manageable combinational auctions. *Management Science*, 44:1131–1147, 1998.

[23] A. Seznec. A new case for skewed-associativity. Technical Report RR-3208, INRIA, Rennes (France), July 1997.

[24] F. Spieksma. On the approximability of an interval scheduling problem. *Journal of Scheduling*, 2:215–227, 1999.

[25] M. Thorup. Undirected single-source shortest paths with positive integer weights in linear time. *Journal of the ACM*, 46:362–394, 1999.

[26] E. Torng. A unified analysis of paging and caching. *Algorithmica*, 20(1):175–200, 1998.

[27] V. Veeramachaneni, P. Berman, and W. Miller. Aligning two fragmented sequences. *Discrete Applied Mathematics*, 2002. To appear.

[28] G. Woeginger. On-line scheduling of jobs with fixed start and end times. *Theoretical Computer Science*, 130:5–16, 1994.

[29] M. Yannakakis and F. Gavril. The maximum k-colorable subgraph problem for chordal graphs. *Information Processing Letters*, 24:133–137, 1987.

A Proof of Theorem 13

In this appendix, we give the proof of Theorem 13. For convenience, we repeat the theorem here.

Theorem 13 *[equal lengths] For $WJISP_2$ with arbitrary weights, $GREEDY_\alpha$ achieves approximation ratio $\frac{2+\alpha}{\alpha(1-\alpha^2)}$.*

Proof: As in the proof of Theorem 10, we distinguish overlap charge and job charge. Some intervals in OPT will create job charge, some will create overlap charge, and some will create both. If an interval $i \in OPT$ creates overlap charge, that overlap charge is denoted by $ov(i)$. The total charge created by an interval $i \in OPT$ is always $w(i)$.

For overlap charge, the charging relation is determined in two steps. First, we define for each interval $i \in OPT$ that creates overlap charge a set $r(i) \subseteq T$ of intervals that are potential targets for the overlap charge of i. For some intervals $i \in OPT$ the set $r(i)$ will be singleton; in this case, we say that the unique interval $j \in r(i)$ receives a *forced overlap charge*. Then we would like to show that each interval $i \in OPT$ with $|r(i)| > 1$ can choose a target interval $k_i \in r(i)$ such that no two intervals in OPT choose the same target. For some intervals $i \in OPT$, however, it will not be possible to find such an interval $k_i \in r(i)$; for such an interval i, we have to argue in a more elaborate way that we can find a collection of fractions of other intervals to which we can charge the overlap charge created by i.

Consider an interval $i \in OPT$ and the instant when GREEDY$_\alpha$ processed that interval. If $i \in T$, charge a forced overlap charge of $ov(i) = w(i)$ to i (i.e., set $r(i) = \{i\}$). Now assume that $i \notin T$. Let Q_i be the set of intervals in S that are in conflict with i when GREEDY$_\alpha$ processes i. Note that Q_i contains at most 2 intervals intersecting i (because S is feasible and all intervals have the same length) and at most one additional interval belonging to the same job as i. Therefore, we have $|Q_i| \leq 3$.

Case 1: Q_i contains an interval i' belonging to the same job as i and overlapping i. Charge a forced overlap charge of $ov(i) = w(i)$ to i' (i.e., set $r(i) = \{i'\}$). Note that $w(i') \geq \alpha w(i)$, since GREEDY$_\alpha$ did not accept i.

Case 2: Q_i does not contain an interval i' belonging to the same job as i and overlapping i. If Q_i contains an interval i' belonging to the same job as i and disjoint from i, charge a job charge of $d_i = \min\{\frac{1}{\alpha}w(i'), w(i)\}$ to i'. Otherwise, let $d_i = 0$. Then, if $w(i) - d_i > 0$, let $r(i) \subseteq S$ be the set of intervals in S that overlap i, i.e., $r(i) = Q_i \setminus \{i'\}$. Note that $r(i)$ contains exactly 2 intervals and that each interval in $r(i)$ has weight at least $\alpha(w(i) - d_i)$, since GREEDY$_\alpha$ did not accept i. The overlap charge created by i is $ov(i) = w(i) - d_i$, and we will try to charge it to an interval $k_i \in r(i)$, where k_i is determined in a second step as described below. Note that k_i can be selected only after all forced overlap charges (of Case 1) have been determined, because otherwise an interval might receive overlap charge twice.

In order to determine the target interval k_i for those intervals $i \in OPT$ with $|r(i)| = 2$, we execute the following greedy procedure:

Procedure ASSIGN
Consider the intervals $i \in OPT$ with $|r(i)| = 2$ in order of nondecreasing left

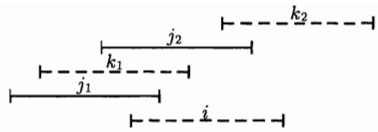


Figure 12: Interval i cannot find a target for its overlap charge, because j_1 and j_2 receive overlap charge from k_1 and k_2. Intervals drawn dashed are in OPT, intervals drawn solid are in T.

> endpoint. For each such interval i, let k_i be an interval in $r(i)$ that has not yet received overlap charge (if such an interval k_i exists). If both intervals in $r(i)$ have not yet received overlap charge, choose the one with smaller right endpoint as k_i. Charge the overlap charge $ov(i)$ created by i to k_i. If no such interval k_i exists (i.e., if both intervals in $r(i)$ have received overlap charge already), add i to a set P of *pending intervals* and, for now, skip the assignment of the overlap charge created by i.

Assume for now that this greedy procedure can assign all overlap charges so that the set P of pending intervals is empty. Then we know that all intervals in OPT charge their weight to intervals in T and that each interval $j \in T$ receives overlap charge at most $w(j)/\alpha$ and job charge at most $w(j)/\alpha$. In order to facilitate our analysis, we redistribute some of the job charge. Consider the case that some interval $j \in T$ receives job charge from an interval $i \in OPT$ belonging to the same job. If $j \in A$, we leave the charge unchanged. If $j \in T \setminus A$, this means that j was preempted by GREEDY$_\alpha$ at some later time in favor of an interval k belonging to the same job as j. Observe that k cannot have received any job charge, as the interval $i \in OPT$ that belongs to the same job as j and k was processed by GREEDY$_\alpha$ before k. Now we redistribute the total charge that j and k have received (job charge and overlap charge), which is bounded from above by $\frac{2}{\alpha}w(j) + \frac{1}{\alpha}w(k)$, proportionally among these two intervals. As in the proof of Theorem 10, we get that j receives charge at most $\left(\frac{1}{\alpha} + \frac{1}{\alpha+1}\right) w(j)$ and k receives charge at most $\left(\frac{1}{\alpha} + \frac{1}{\alpha+1}\right) w(k)$. In this way every interval $j \in A$ receives charge at most $\frac{2}{\alpha}w(j)$, while every interval $k \in T \setminus A$ receives charge at most $(\frac{1}{\alpha} + \frac{1}{\alpha+1})w(k)$. As shown in the proof of Theorem 10, this implies $w(OPT) \leq w(A)\frac{2+\alpha}{\alpha(1-\alpha^2)}$.

It remains to deal with the case that $P \neq \emptyset$. Our goal is to charge the overlap charge created by the intervals in P to (fractions of) intervals in T such that each interval $i \in A$ receives total charge at most $\frac{2}{\alpha}w(i)$ and each interval $j \in T \setminus A$ receives total charge at most $(\frac{1}{\alpha} + \frac{1}{\alpha+1})w(j)$. Once we have accomplished this, the same analysis as in the case that the set P of pending intervals is empty can establish the claimed bound on the approximation ratio of GREEDY$_\alpha$.

In the following, we write $i^{(\ell)}$ and $i^{(r)}$ for the left and right endpoint of the interval i, respectively. Consider some interval $i \in P$. As i was added to P by procedure ASSIGN, both intervals in $r(i)$ were already targets of overlap charge when i was processed by ASSIGN. This means that we must have the configuration depicted in Figure 12: i contains the right endpoints of two intervals $j_1, j_2 \in T$, and exactly one of them, say j_2, has received a forced overlap charge from an interval $k_2 \in OPT$ satisfying $k_2^{(r)} > i^{(r)}$ and belonging to the same job as j_2. j_1 must have received overlap charge from an interval $k_1 \in OPT$ with $j_1^{(r)} \leq k_1^{(r)} < k_2^{(\ell)}$.

There may be intervals in T with right endpoints between $i^{(r)}$ and $k_2^{(r)}$, but these are not shown in Figure 12. We call the set $\{i, j_1, k_1, j_2, k_2\}$ a *bad configuration*. Notice that interval k_1 may be identical to interval j_1. Observe that neither k_1 nor k_2 can be in P, because both of them have found a target for their overlap charge. Therefore, the intervals in P are pairwise disjoint. Furthermore, no other interval in P can overlap k_1 or k_2.

For each bad configuration, we will start at the interval $i \in P$ contained in that configuration and scan intervals either to the right or to the left in order to find (fractions of) intervals that can become targets of the overlap charge created by i. The process will be such that no interval can be visited during the scans starting at two different bad configurations.

We let $A' \subseteq T$ denote the set of intervals in T that are either in A or were preempted by GREEDY$_\alpha$ when a *disjoint* interval belonging to the same job was processed. A *chain* is a maximal sequence $\langle i_1, i_2, \ldots, i_\ell \rangle$ of intervals in T such that the following conditions hold:

- i_1 was selected by GREEDY$_\alpha$ without preempting an interval intersecting i_1.
- For $2 \leq j \leq \ell$, interval i_j overlaps i_{j-1} and GREEDY$_\alpha$ preempted i_{j-1} when it selected i_j.

By this definition, T is partitioned into chains in a unique way. An interval in T can only receive job charge if it is still in S when a disjoint interval to the right is processed. This can happen only for the last interval of a chain. The set of the last intervals of all chains is just the set A'.

Consider a bad configuration with intervals i, j_1, j_2, k_1, k_2 (cf. Figure 12). Note that j_2 cannot receive any job charge or redistributed job charge, as the interval in OPT that belongs to the same job as j_2 is k_2 (so that no job charge is created for this job at all). Therefore, we can charge $\frac{1}{\alpha+1}w(j_2)$ additional charge to j_2 and still get that j_2 receives a total charge of at most $(\frac{1}{\alpha} + \frac{1}{\alpha+1})w(j_2)$, which is sufficient for our analysis. Recall that $ov(i)$ is the overlap charge created by i. As $w(j_2) \geq \alpha ov(i)$, the overlap charge created by i that remains after charging $\frac{1}{\alpha+1}w(j_2)$ to j_2 is at most $r_i = ov(i) - \frac{1}{\alpha+1}w(j_2) \leq \frac{1}{\alpha+1}ov(i)$. We must find (fractions of) intervals to which we can charge r_i. We distinguish the following cases.

Case 1: k_1 belongs to the same job as j_1.

In this case, we will find the targets for r_i either within the bad configuration containing i or to the right of it.

Case 1.1: $j_1 \in A'$.

Note that j_1 cannot receive any job charge or redistributed job charge, since the interval in OPT belonging to the same job is k_1. Therefore, j_1 can accept up to $\frac{1}{\alpha}w(j_1)$ additional charge: if $j_1 \in A$, this is fine, and if $j_1 \in A' \setminus A$, the charge received by j_1 can be redistributed among j_1 and the interval of the same job that preempts j_1. As $r_i \leq ov(i) \leq \frac{1}{\alpha}w(j_1)$, we can charge r_i to j_1 in this case.

Case 1.2: $j_1 \notin A'$.

Let y be the interval that preempts j_1, i.e., the successor of j_1 in its chain. We must have $i^{(r)} \leq y^{(r)} \leq k_2^{(r)}$. Note that $w(y) \geq \frac{1}{\alpha}w(j_1)$ and that $r_i \leq \frac{1}{\alpha+1}ov(i) \leq \frac{1}{\alpha+1} \cdot \frac{1}{\alpha}w(j_1) \leq \frac{1}{\alpha+1}w(y)$. We distinguish three subcases with respect to the overlap charge received by y.

Figure 13: Recursive argument of Case 1.2.3: The configuration of j_2, i, y, k_2 (left-hand side) is the same as that of y, k_2, x, y' (right-hand side). Each configuration consists of four intervals g_1, g_2, g_3, g_4 with the following properties. g_1 and g_3 are from T, and g_2 and g_4 are from OPT. We have $g_1^{(r)} \leq g_2^{(r)} \leq g_3^{(r)} \leq g_4^{(r)}$ and $g_2^{(r)} < g_4^{(r)}$. g_1 overlaps g_4. After ASSIGN is executed, g_4 has assigned overlap charge to g_1. If $g_4 \neq k_2$, then the intervals g_1 and g_3 were in S when g_4 was processed and do not belong to the same job as g_4. The remaining overlap charge (that is to be distributed during the current scan to the right of a bad configuration, before additional charge is assigned to g_3) is at most $\frac{1}{\alpha+1}w(g_3)$.

Case 1.2.1: y does not receive any overlap charge.

Since intervals in T can accept overlap charge up to $\frac{1}{\alpha}$ times their weight, we can charge r_i to y.

Case 1.2.2: y receives overlap charge from an interval of the same job.

In this case, y cannot receive any job charge or redistributed job charge. Therefore, y can accept $\frac{1}{\alpha+1}w(y)$ additional charge. Since $r_i \leq \frac{1}{\alpha+1}w(y)$, we can charge r_i to y.

Case 1.2.3: y receives overlap charge from an interval of a different job.

Let $y' \in OPT$ be the interval that assigns overlap charge to y. We must have $y'^{(\ell)} \leq y^{(r)} < y'^{(r)}$. At the time when GREEDY$_\alpha$ processed y', S must have contained a second interval x (besides y) intersecting y'. Note that $w(x) \geq \alpha ov(y')$. As j_2 is in S when k_2 is processed, we must have $x^{(r)} \geq k_2^{(r)}$.

As y can accept $\frac{1}{\alpha}w(y)$ total overlap charge, it can accept $\frac{1}{\alpha}w(y) - ov(y')$ additional charge if $ov(y') < \frac{1}{\alpha}w(y)$. We charge up to $\frac{1}{\alpha}w(y) - ov(y')$ of r_i to y. Then the remaining charge r_i' is at most $\frac{1}{\alpha+1}w(x)$: If $w(x) \geq w(y)$, this is obvious, since we have $r_i' \leq r_i$ and $r_i \leq \frac{1}{\alpha+1}w(y)$. If $w(x) < w(y)$, we can calculate as follows:

$$\begin{aligned} r_i' &= r_i - (\frac{1}{\alpha}w(y) - ov(y')) \\ &\leq \frac{1}{\alpha+1}w(y) - \frac{1}{\alpha}w(y) + \frac{1}{\alpha}w(x) \\ &= -\frac{1}{\alpha(\alpha+1)}w(y) + \frac{1}{\alpha}w(x) \\ &\leq -\frac{1}{\alpha(\alpha+1)}w(x) + \frac{1}{\alpha}w(x) = \frac{1}{\alpha+1}w(x) \end{aligned}$$

Now the situation for r_i' and x is exactly the same as it was for r_i and y, see Figure 13. Therefore, we can apply the arguments of Case 1.2 to r_i' and x in place of r_i and y recursively. Since the set of intervals is finite, the recursion must terminate.

Case 2: k_1 does not belong to the same job as j_1.

In this case, we will find the targets for r_i either within the bad configuration containing i or to the left of it. At the time when k_1 was processed, S must have contained a second

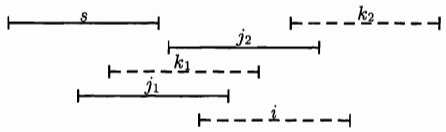


Figure 14: Illustration of Case 2.

interval (in addition to j_1) overlapping k_1. Call this interval s and note that $w(s) \geq \alpha ov(k_1)$. See Figure 14. We have $r_i \leq \frac{1}{\alpha+1} ov(i) \leq \frac{1}{\alpha(\alpha+1)} w(j_1)$. If $ov(k_1) < \frac{1}{\alpha} w(j_1)$, we charge additional $\frac{1}{\alpha} w(j_1) - ov(k_1)$ to j_1, because j_1 can take up to $\frac{1}{\alpha} w(j_1)$ overlap charge. Let r'_i denote the remaining charge, i.e., $r'_i = r_i - \left(\frac{1}{\alpha} w(j_1) - ov(k_1)\right)$. We claim that $r'_i \leq \frac{1}{\alpha(\alpha+1)} w(s)$. If $w(s) \geq w(j_1)$, this is obvious. If $w(s) < w(j_1)$, we can calculate as follows:

$$\begin{aligned} r'_i &= r_i - \left(\frac{1}{\alpha} w(j_1) - ov(k_1)\right) \\ &\leq \frac{1}{\alpha(\alpha+1)} w(j_1) - \frac{1}{\alpha} w(j_1) + \frac{1}{\alpha} w(s) \\ &= -\frac{1}{\alpha+1} w(j_1) + \frac{1}{\alpha} w(s) \\ &\leq -\frac{1}{\alpha+1} w(s) + \frac{1}{\alpha} w(s) = \frac{1}{\alpha(\alpha+1)} w(s) \end{aligned}$$

In the case that s is *not* the last interval of its chain, we can get the tighter bound $r'_i \leq \frac{1}{\alpha+1} w(s)$ using the following arguments. As s is still in S when k_1 is processed, s must belong to the same chain as j_2 and we must have $w(s) \leq \alpha w(j_2)$. Recall that we have charged $\frac{1}{\alpha+1} w(j_2)$ of $ov(i)$ to j_2. Furthermore, j_1 can take $\frac{1}{\alpha} w(j_1) - ov(k_1)$ additional charge, since intervals in T can accept $\frac{1}{\alpha}$ times their weight as overlap charge. So we can calculate as follows:

$$\begin{aligned} r'_i &= ov(i) - \frac{1}{\alpha+1} w(j_2) - \left(\frac{1}{\alpha} w(j_1) - ov(k_1)\right) \\ &\leq \frac{1}{\alpha} w(j_1) - \frac{1}{\alpha+1} w(j_2) - \frac{1}{\alpha} w(j_1) + \frac{1}{\alpha} w(s) \\ &\leq -\frac{1}{\alpha(\alpha+1)} w(s) + \frac{1}{\alpha} w(s) = \frac{1}{\alpha+1} w(s) \end{aligned}$$

Case 2.1: s is the last interval of its chain, i.e., $s \in A'$.
As shown above, the remaining overlap charge created by i is $r'_i \leq \frac{1}{\alpha(\alpha+1)} w(s)$.

Case 2.1.1: s does not receive any overlap charge.
This case cannot occur. If s is the last interval of its chain, it must end before $j_2^{(\ell)}$ and, therefore, before $j_1^{(r)}$. But then procedure ASSIGN would have chosen s instead of j_1 as the target for the overlap charge of k_1, a contradiction.

Case 2.1.2: s receives overlap charge from an interval of the same job.
In this case, s does not receive any job charge or redistributed job charge. Therefore,

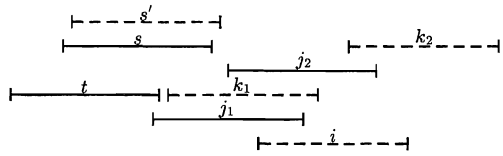


Figure 15: Illustration of Case 2.1.3.

we can charge $r'_i \leq \frac{1}{\alpha}w(s)$ to s. If $s \in A$, this is fine. If $s \in A' \setminus A$, the charge of s can be redistributed between s and the disjoint interval of the same job that preempts s, in the usual way.

Case 2.1.3: s receives overlap charge from an interval of a different job.
Denote that interval by s'. We must have $s^{(r)} \leq s'^{(r)} < i^{(\ell)}$. At the time when GREEDY$_\alpha$ processed s', S must have contained a second interval t overlapping s'. See Figure 15. If $ov(s') < \frac{1}{\alpha}w(s)$, we charge $\frac{1}{\alpha}w(s) - ov(s')$ of r'_i to s. The remaining overlap charge is $r''_i = r'_i - (\frac{1}{\alpha}w(s) - ov(s'))$.

Case 2.1.3.1: t is the last interval of its chain.
We claim that $r''_i \leq \frac{1}{\alpha(\alpha+1)}w(t)$. If $w(t) \geq w(s)$, this is obvious. If $w(t) < w(s)$, calculate as follows:

$$\begin{aligned} r''_i &= r'_i - (\frac{1}{\alpha}w(s) - ov(s')) \\ &\leq \frac{1}{\alpha(\alpha+1)}w(s) - \frac{1}{\alpha}w(s) + \frac{1}{\alpha}w(t) \\ &= -\frac{1}{\alpha+1}w(s) + \frac{1}{\alpha}w(t) \\ &\leq -\frac{1}{\alpha+1}w(t) + \frac{1}{\alpha}w(t) = \frac{1}{\alpha(\alpha+1)}w(t) \end{aligned}$$

Therefore, we can apply the arguments of Case 2 recursively to r''_i and t in place of r'_i and s (cf. Figure 16).

Case 2.1.3.2: t is not the last interval of its chain.
As t is still in S when s' is processed, t must belong to the same chain as j_1 and we must have $w(t) \leq \alpha w(j_1)$. Recall that the remaining charge was $r_i \leq \frac{1}{\alpha(\alpha+1)}w(j_1)$ initially, before we charged any additional charge to j_1. Then we have charged $\frac{1}{\alpha}w(j_1) - ov(k_1)$ to j_1. Furthermore, s can take $\frac{1}{\alpha}w(s) - ov(s')$ additional charge, since intervals in T can accept $\frac{1}{\alpha}$ times their weight as overlap charge. Let r''_i denote the remaining charge after charging these amounts to j_1 and s. We can calculate as follows:

$$\begin{aligned} r''_i &\leq \frac{1}{\alpha(\alpha+1)}w(j_1) - \left(\frac{1}{\alpha}w(j_1) - ov(k_1)\right) - \left(\frac{1}{\alpha}w(s) - ov(s')\right) \\ &\leq \frac{1}{\alpha(\alpha+1)}w(j_1) - \frac{1}{\alpha}w(j_1) + \frac{1}{\alpha}w(s) - \frac{1}{\alpha}w(s) + \frac{1}{\alpha}w(t) \\ &\leq -\frac{1}{\alpha+1}w(j_1) + \frac{1}{\alpha}w(t) \end{aligned}$$

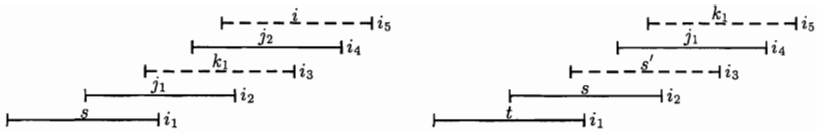


Figure 16: Recursive argument of Case 2: The configuration of s, j_1, k_1, j_2, i (left-hand side) is the same as that of t, s, s', j_1, k_1 (right-hand side). Each configuration consists of five intervals that can be named i_1, i_2, i_3, i_4, i_5 such that the following properties hold. i_1, i_2 and i_4 are from T, and i_3 and i_5 are from OPT. i_1 and i_2 overlap i_3 and were in S when i_3 was processed by GREEDY$_\alpha$. i_2 and i_4 overlap i_5 and were in S when i_5 was processed. We have $i_3^{(r)} < i_4^{(r)} \leq i_5^{(r)}$. i_3 has assigned overlap charge to i_2 in procedure ASSIGN. Let r denote the remaining overlap charge that is to be assigned during the current scan to the left of a bad configuration before assigning additional charge to i_1 and i_2. Then r is at most $\frac{1}{\alpha(\alpha+1)}w(i_2)$. If $i_1 \in A'$, then $r - (\frac{1}{\alpha}w(i_2) - ov(i_3)) \leq \frac{1}{\alpha(\alpha+1)}w(i_1)$. If $i_1 \notin A'$, then $r - (\frac{1}{\alpha}w(i_2) - ov(i_3)) \leq \frac{1}{\alpha+1}w(i_1)$.

$$\leq -\frac{1}{\alpha(\alpha+1)}w(t) + \frac{1}{\alpha}w(t) = \frac{1}{\alpha+1}w(t)$$

Therefore, we can apply the arguments of Case 2 recursively to r_i'' and t in place of r_i' and s (cf. Figure 16).

Case 2.2: s is not the last interval of its chain.
As shown above, the remaining overlap charge created by i is $r_i' \leq \frac{1}{\alpha+1}w(s)$ in this case.

Case 2.2.1: s does not receive any overlap charge.
Since intervals in T can accept overlap charge up to $\frac{1}{\alpha}$ times their weight and since $r_i' \leq \frac{1}{\alpha+1}w(s) < \frac{1}{\alpha}w(s)$, we can charge r_i' to s. Observe that we must have $s^{(r)} \geq j_1^{(r)}$ in this case, because otherwise procedure ASSIGN would have chosen s as the target for the overlap charge created by k_1.

Case 2.2.2: s receives overlap charge from an interval of the same job.
In this case, s cannot receive any job charge or redistributed job charge. Therefore, s can accept $\frac{1}{\alpha+1}w(s)$ additional charge. Since $r_i' \leq \frac{1}{\alpha+1}w(s)$, we can charge r_i' to s.

Case 2.2.3: s receives overlap charge from an interval of a different job.
Denote that interval by s'. We must have $s^{(r)} \leq s'^{(r)} < i^{(\ell)}$. At the time when GREEDY$_\alpha$ processed s', S must have contained a second interval t overlapping s'. See again Figure 15. If $ov(s') < \frac{1}{\alpha}w(s)$, we charge $\frac{1}{\alpha}w(s) - ov(s')$ of r_i' to s. The remaining overlap charge is $r_i'' = r_i' - (\frac{1}{\alpha}w(s) - ov(s'))$. We claim that $r_i'' \leq \frac{1}{\alpha+1}w(t)$. If $w(t) \geq w(s)$, this is obvious. If $w(t) < w(s)$, calculate as follows:

$$\begin{aligned} r_i'' &= r_i' - (\frac{1}{\alpha}w(s) - ov(s')) \\ &\leq \frac{1}{\alpha+1}w(s) - \frac{1}{\alpha}w(s) + \frac{1}{\alpha}w(t) \\ &= -\frac{1}{\alpha(\alpha+1)}w(s) + \frac{1}{\alpha}w(t) \end{aligned}$$

$$\leq -\frac{1}{\alpha(\alpha+1)}w(t) + \frac{1}{\alpha}w(t) = \frac{1}{\alpha+1}w(t)$$

Therefore, we can apply the arguments of Case 2 recursively to r_i'' and t in place of r_i' and s (cf. Figure 16).

For every interval $i \in P$, the above case analysis shows that we can find (fractions of) intervals in T to which we can charge $ov(i)$ without violating the condition that each interval $j \in A$ receives charge at most $\frac{2}{\alpha}w(j)$ and each interval $j \in T \setminus A$ receives charge at most $(\frac{1}{\alpha} + \frac{1}{\alpha+1})w(j)$.

It remains to show that no interval receives additional charge from two different intervals $i, i' \in P$ $(i \neq i')$. Assume to the contrary that this happens. Then one of the following cases must occur. In each case, this leads to a contradiction. We name the intervals visited during a scan to the right or to the left as shown in Figures 13 and 16.

- The scan starting at $i \in P$ and going to the right hits another bad configuration. Since a scan to the right meets only configurations as those shown in Figure 13, this cannot happen: In the configuration after a step to the right, the interval named g_4 in Figure 13 cannot be part of another bad configuration. This can be proved as follows: Assume to the contrary that g_4 is the first interval in OPT visited during a scan to the right that belongs to a different bad configuration. Since g_1 has received overlap charge from g_4, g_4 cannot be the interval $i \in P$ of a different bad configuration (cf. Figure 12). If g_4 were the interval k_2 of a different bad configuration, the interval g_2 would be the interval $i \in P$ of that configuration, contradicting the choice of g_4. If g_4 were the interval k_1 of a different bad configuration, the interval g_1 would have to be the interval j_1 of that configuration, but this is impossible because then the right endpoint of g_1 would have to be contained in an interval in P (which would intersect g_2 and g_4).

  So we know that g_4 is not part of another bad configuration. But then g_3 cannot be the interval j_1 or j_2 of another bad configuration either, because then the right endpoint of g_3 would have to be contained in two intervals from OPT belonging to that bad configuration. Thus, a scan to the right cannot hit another bad configuration.

- The scan starting at $i \in P$ and going to the left hits another bad configuration. A scan to the left meets only configurations as those shown in Figure 16. After a step to the left, the newly considered interval $i_3 \in OPT$ cannot be part of another bad configuration. This can be verified in a similar way as in the previous case. Therefore, the interval $i_1 \in T$ cannot be the interval j_1 or j_2 of another bad configuration either.

- The scan starting at $i \in P$ going to the right hits the scan starting at $i' \in P$ going to the left. After a step to the right during the scan starting at $i \in P$ (cf. Figure 13), the interval $g_4 \in OPT$ is such that there were two intervals overlapping g_4 in S when g_4 was processed by GREEDY$_\alpha$ and the overlap charge of g_4 is not assigned to the one with larger right endpoint. During a scan going to the left (cf. Figure 16), if the current step is not the last one, then the interval $i_3 \in OPT$ has $|r(i_3)| = 2$ and has assigned its overlap charge (during the execution of ASSIGN) to the interval in $r(i_3)$ with strictly larger right endpoint. Therefore, the interval i_3 in a scan to the left (except in the last step) can never be equal to the interval g_4 in a scan to the right. So the intervals in OPT visited during a scan to the left (except in the last step) are disjoint from those visited during a scan to the right. Consequently, the intervals that receive additional

charge during a scan to the left (except in the last step) and those that receive additional charge during a scan to the right are also different.

It remains to consider the possibility that the interval i_1 of the last step during a scan to the left is equal to the interval g_3 of a configuration visited during a scan to the right. In that case, $i_1 = g_3$ must either have received no overlap charge initially (Case 1.2.1 for the scan to the right, Case 2.2.1 for the scan to the left) or it must have received overlap charge from an interval belonging to the same job (Case 1.2.2, Cases 2.1.2 or 2.2.2).

- If $i_1 = g_3$ has not received any overlap charge initially, the scan to the left must have ended in Case 2.2.1. There we have observed that $i_1^{(r)} \geq i_2^{(r)}$ in this case. Then $i_1^{(r)}$ is contained in $i_3 \in OPT$ and in $i_5 \in OPT$ (see Figure 16). Furthermore, $g_3^{(r)} = i_1^{(r)}$ is contained in $g_4 \in OPT$ (see Figure 13). Therefore, we must have $g_4 = i_3$ or $g_4 = i_5$.
  If $g_4 = i_3$, we must have $g_1 = i_2$, because g_4 has assigned overlap charge to g_1, and i_3 has assigned overlap charge to i_2. But then $g_2 = i_5$, because the right endpoint of $g_1 = i_2$ is contained in intervals g_2, g_4, i_3, i_5. This is impossible, because $g_2^{(r)} < g_4^{(r)}$ and $i_3^{(r)} < i_5^{(r)}$.
  Now consider the case that $g_4 = i_5$. The interval i_5 is either equal to an interval of the bad configuration at which the scan to the left has started or it was the interval i_3 of the previous step in the scan to the left. The former is impossible because a scan to the right cannot hit another bad configuration, the latter is impossible because we have already shown that the intervals in OPT visited during a scan to the left (except the last step) are disjoint from those visited in a scan to the right.
- If $i_1 = g_3$ has received overlap charge from an interval belonging to the same job, we must have $i_1^{(r)} < i_2^{(r)}$, see Figure 16. This implies that the right endpoint of i_1 is contained in i_3, in g_4, and in the interval belonging to the same job. Since i_3 and g_4 do not belong to the same job as i_1 (they have not assigned overlap charge to i_1), we must have $i_3 = g_4$. But then we must have $g_1 = i_2$. This is impossible, because $g_1^{(r)} < g_3^{(r)}$ (see Figure 13) but $i_1^{(r)} < i_2^{(r)}$.

Therefore, the scans starting at different bad configurations can never assign additional charge to the same interval. This completes the proof. □